



APUSIC
固若长城
睿比世界

Migration Manual

金蝶Apusic分布式消息队列V2.0.6_for_rabbitmq

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 Migration Overview
 - 2.1 Migration Principles
 - 2.2 Migration Scope
 - 2.3 Pre-Migration Checks
 - 2.3.1 Version Compatibility Check
 - 2.3.2 Client Compatibility Check
- 3 Migration Plans
 - 3.1 Plan 1: Dual-Write Migration (Recommended)
 - 3.2 Plan 2: Downtime Migration
 - 3.3 Plan 3: Grayscale Migration
- 4 Configuration Migration
 - 4.1 Export Original RabbitMQ Configuration
 - 4.2 Create Configuration in ADMQ for RabbitMQ
 - 4.2.1 Using the Console
 - 4.2.2 Using the Command Line
- 5 Data Migration
 - 5.1 Migrating Data Using Shovel
 - 5.1.1 Configuring Shovel
 - 5.1.2 Creating Shovel Configuration
 - 5.2 Migrating Data Using Federation
 - 5.3 Migrating Data Using Client Scripts
- 6 Client Migration
 - 6.1 Connection Address Change
 - 6.2 SSL/TLS Configuration Changes
 - 6.3 Common Client Configurations
 - 6.3.1 Java (Spring Boot)
 - 6.3.2 Python (Django)
 - 6.3.3 Go

- 7 Migration Verification
 - 7.1 Functional Verification
 - 7.2 Performance Verification
 - 7.3 Data Consistency Verification
- 8 Rollback Plan
 - 8.1 Rollback Checklist
- 9 Migration Best Practices

1 Preface

This document is the migration guide for Kingdee Apusic Distributed Message Queue for RabbitMQ (Apusic Distributed Message Queue, referred to as: ADMQ for RabbitMQ), detailing how to migrate from RabbitMQ to ADMQ for RabbitMQ.

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 RabbitMQ products, please refer to the following ADMQ for RabbitMQ product documentation set:

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

8	Kingdee Apusic Distributed Message Queue for RabbitMQ Performance Optimization Guide	Detailed description of ADMQ for RabbitMQ performance tuning.
---	--	---

1.3 Technical Support

ADMQ for RabbitMQ 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 seeking technical support, please provide the following information:

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

2 Migration Overview

2.1 Migration Principles

ADMQ for RabbitMQ is fully compatible with the RabbitMQ AMQP 0-9-1 protocol. The migration process follows these principles:

1. **Zero code changes:** Client code does not need modification, only the connection address needs to be changed
2. **Configuration compatibility:** Supports native RabbitMQ configuration parameters
3. **Data migratable:** Supports migration of queue data and configuration information
4. **Smooth switchover:** Supports dual-write and grayscale switchover to reduce migration risk

2.2 Migration Scope

Migration Item	Supported	Description
Application code	Yes	No modification needed, only connection address change required
Client configuration	Yes	Supports native RabbitMQ clients
Exchange definition	Yes	Supports all exchange types
Queue definition	Yes	Supports Classic/Quorum/Stream queues
Binding relationships	Yes	Supports all binding relationships
User permissions	Yes	Supports user and Vhost permission migration
Message data	Yes	Supports historical message migration
Policy	Yes	Supports mirroring policies, etc.
Plugins	Partial	Some plugins may require adaptation

2.3 Pre-Migration Checks

2.3.1 Version Compatibility Check

Original RabbitMQ Version	Compatibility	Description
3.8.x	Fully compatible	Recommended migration version
3.9.x	Fully compatible	Recommended migration version

3.10.x	Fully compatible	Recommended migration version
3.11.x	Fully compatible	Recommended migration version
3.12.x	Fully compatible	Recommended migration version

2.3.2 Client Compatibility Check

Client Type	Compatible Version	Description
Java (amqp-client)	5.x	Fully compatible
Python (pika)	1.x	Fully compatible
Go (amqp091-go)	1.x	Fully compatible
Node.js (amqplib)	0.10.x	Fully compatible
.NET (RabbitMQ.Client)	6.x	Fully compatible

3 Migration Plans

3.1 Plan 1: Dual-Write Migration (Recommended)

Suitable for business scenarios that cannot tolerate downtime, ensuring data consistency through dual-write.

Steps:

1. Deploy ADMQ for RabbitMQ cluster

- Deploy the new cluster according to the installation guide
- Configure network and firewall settings

2. Configure dual-write

- Connect to both the original RabbitMQ and ADMQ for RabbitMQ in the application simultaneously
- Send messages to both clusters when publishing

3. Sync historical data

- Use Shovel or Federation plugins to synchronize queue data
- Verify data consistency

4. Switch consumers

- First switch some consumers to ADMQ for RabbitMQ
- Observe whether the business operates normally

5. Full switchover

- After confirmation, switch all consumers
- Stop sending messages to the original RabbitMQ

6. Decommission original cluster

- Keep it for a period for data queries
- Decommission after confirming no issues

3.2 Plan 2: Downtime Migration

Suitable for business scenarios that can tolerate brief downtime, with faster migration speed.

Steps:

1. Stop producers

- Pause the business system from sending messages to RabbitMQ

2. Export configuration

- Export Exchange, Queue, Binding, User and other configurations using `bin/admq rabbitmq admin ctl`

3. Migrate configuration to ADMQ for RabbitMQ

- Create corresponding resources via the console or command line

4. Migrate message data

- Use shovel to export original queue messages
- Import into ADMQ for RabbitMQ

5. Switch connection address

- Modify application configuration to point to ADMQ for RabbitMQ

6. Start producers

- Resume business system message sending

3.3 Plan 3: Grayscale Migration

Suitable for large-scale systems, gradually migrating different business modules.

Steps:

1. Divide migration batches by business module
2. Execute each batch according to the dual-write migration plan
3. Proceed to the next batch after verification
4. Decommission the original cluster after all batches are complete

4 Configuration Migration

4.1 Export Original RabbitMQ Configuration

```
# 导出用户和权限
bin/admq rabbitmq admin ctl list_users > users.txt
bin/admq rabbitmq admin ctl list_vhosts > vhosts.txt

# 导出 Exchange
bin/admq rabbitmq admin ctl list_exchanges name type durable
auto_delete internal arguments > exchanges.txt

# 导出 Queue
bin/admq rabbitmq admin ctl list_queues name durable auto_delete
arguments pid slave_pids > queues.txt

# 导出 Binding
bin/admq rabbitmq admin ctl list_bindings source_name source_kind
destination_name destination_kind routing_key arguments >
bindings.txt

# 导出策略
bin/admq rabbitmq admin ctl list_policies > policies.txt
```

4.2 Create Configuration in ADMQ for RabbitMQ

4.2.1 Using the Console

1. Log in to the console
2. Navigate to the "Resource Management" page
3. Create in order:
 - Vhost
 - MQ users and permissions
 - Exchange
 - Queue

- Binding

4.2.2 Using the Command Line

```
# 创建 Vhost
bin/admq rabbitmq admin ctl add_vhost /test

# 创建用户
bin/admq rabbitmq admin ctl add_user testuser testpass
bin/admq rabbitmq admin ctl set_user_tags testuser management
bin/admq rabbitmq admin ctl set_permissions -p /test testuser ".*"
".*" ".*"

# 创建 Exchange
bin/admq rabbitmq admin ctl eval
'rabbit_exchange:declare(<<"/test">>, <<"test.exchange">>,
<<"direct">>, true, false, false, [], <<"acting_user">>).'

# 创建 Queue
bin/admq rabbitmq admin ctl eval
'rabbit_amqqueue:declare(<<"/test">>, <<"test.queue">>, true, false,
false, [], none, <<"acting_user">>).'
```

5 Data Migration

5.1 Migrating Data Using Shovel

Shovel is a built-in data migration tool in RabbitMQ.

5.1.1 Configuring Shovel

```
# 启用 shovel 插件
rabbitmq-plugins enable rabbitmq_shovel
rabbitmq-plugins enable rabbitmq_shovel_management
```

5.1.2 Creating Shovel Configuration

```
{
  "shovels": [
    {
      "name": "migrate-queue1",
      "source": {
        "uri": "amqp://guest:guest@old-rabbitmq:5672/%2F",
        "queue": "queue1"
      },
      "destination": {
        "uri": "amqp://guest:guest@adm-q-rabbitmq:5672/%2F",
        "queue": "queue1"
      },
      "ack-mode": "on-confirm"
    }
  ]
}
```

5.2 Migrating Data Using Federation

Federation is suitable for cross-cluster data synchronization.

```
# 启用 federation 插件
rabbitmq-plugins enable rabbitmq_federation
rabbitmq-plugins enable rabbitmq_federation_management
```

5.3 Migrating Data Using Client Scripts

For small-scale data, you can write client scripts for migration:

```
import pika

# 连接原 RabbitMQ
src_conn = pika.BlockingConnection(pika.ConnectionParameters('old-
rabbitmq'))
src_ch = src_conn.channel()

# 连接 ADMQ for RabbitMQ
dst_conn = pika.BlockingConnection(pika.ConnectionParameters('admq-
rabbitmq'))
dst_ch = dst_conn.channel()

# 获取消息并转发
while True:
    method, properties, body = src_ch.basic_get('queue1',
auto_ack=False)
    if method is None:
        break

    dst_ch.basic_publish('', 'queue1', body, properties)
    src_ch.basic_ack(method.delivery_tag)

src_conn.close()
dst_conn.close()
```

6 Client Migration

6.1 Connection Address Change

Only the connection address in the application configuration file needs to be modified:

```
# 原配置
rabbitmq.host=old-rabbitmq
rabbitmq.port=5672

# 新配置
rabbitmq.host=admq-rabbitmq
rabbitmq.port=5672
```

6.2 SSL/TLS Configuration Changes

If SSL is enabled, certificates need to be updated:

```
# 原配置
rabbitmq.ssl.enabled=true
rabbitmq.ssl.ca-file=/path/to/old-ca.pem

# 新配置
rabbitmq.ssl.enabled=true
rabbitmq.ssl.ca-file=/path/to/admq-ca.pem
rabbitmq.ssl.cert-file=/path/to/client-cert.pem
rabbitmq.ssl.key-file=/path/to/client-key.pem
```

6.3 Common Client Configurations

6.3.1 Java (Spring Boot)

```
spring:
  rabbitmq:
```

```
host: admq-rabbitmq
port: 5672
username: guest
password: guest
virtual-host: /
ssl:
  enabled: false
```

6.3.2 Python (Django)

```
BROKER_URL = 'amqp://guest:guest@admq-rabbitmq:5672/'
```

6.3.3 Go

```
conn, err := amqp091.Dial("amqp://guest:guest@admq-rabbitmq:5672/")
```

7 Migration Verification

7.1 Functional Verification

Verification Item	Verification Method
Connection test	Use a client to connect to ADMQ for RabbitMQ and confirm the connection is established successfully
Message sending	Send test messages and confirm they reach the Exchange successfully
Message routing	Verify messages are correctly routed to the Queue
Message consumption	Start a consumer and confirm messages can be received and processed normally
Message acknowledgment	Test manual and automatic acknowledgment functions
Persistence	Verify messages are retained after service restart
Cluster functionality	Verify automatic failover when a node fails

7.2 Performance Verification

Verification Item	Verification Method
Throughput	Compare message send rates between the original cluster and the new cluster
Latency	Measure the delay from message send to consumption
Concurrency	Test multi-producer, multi-consumer concurrent scenarios
Stability	Run long-duration tests and observe for any anomalies

7.3 Data Consistency Verification

```
# 对比原集群和新集群的队列消息数
bin/admq rabbitmq admin ctl list_queues -p /test name messages

# 对比消费者数量
bin/admq rabbitmq admin ctl list_queues -p /test name consumers
```

对比连接数

```
bin/admq rabbitmq admin ctl list_connections -p /test peer_host  
state
```

8 Rollback Plan

If issues arise during the migration process, roll back to the original RabbitMQ cluster:

1. Stop sending messages to ADMQ for RabbitMQ
2. Switch the connection address back to the original RabbitMQ
3. Sync new messages from ADMQ for RabbitMQ back to the original cluster (if needed)
4. Restore the business system

8.1 Rollback Checklist

- ☐ Original RabbitMQ cluster is still running
- ☐ Original cluster data has not been deleted
- ☐ Application configuration can be switched quickly
- ☐ Business system supports fast rollback

9 Migration Best Practices

1. **Thorough testing:** Complete the full migration process validation in a test environment
2. **Backup data:** Back up the original cluster configuration and data before migration
3. **Monitor the migration process:** Monitor various metrics in real-time during migration
4. **Prepare a rollback plan:** Ensure quick rollback in case of issues
5. **Choose off-peak hours:** Perform migration during business off-peak hours
6. **Gradual switchover:** Switch non-core business first, then core business after verification
7. **Retain original cluster:** Keep the original cluster for a period after migration for reference

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

