



Migration Manual

Kingdee Apusic Distributed Message Queue
V2.0.6_for_mqtt

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 Checklist
 - 2.3.1 Protocol 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: Gray-Scale Migration
- 4 Configuration Migration
 - 4.1 User Migration
 - 4.1.1 Export Users from Original Broker
 - 4.1.2 Create Users in ADMQ for MQTT
 - 4.2 ACL Rule Migration
 - 4.2.1 Export ACL Rules from Original Broker
 - 4.2.2 Create ACL Rules in ADMQ for MQTT
 - 4.3 Listener Migration
 - 4.3.1 Export Listener Configuration from Original Broker
 - 4.3.2 Create Listeners in ADMQ for MQTT
- 5 Retained Message Migration
 - 5.1 Migration Using Client Scripts
- 6 Client Migration
 - 6.1 Connection Address Change
 - 6.2 SSL/TLS Configuration Change
 - 6.3 Common Client Configuration
 - 6.3.1 Java (Spring Boot)
 - 6.3.2 Python

- 6.3.3 JavaScript
- 7 Migration Verification
 - 7.1 Functional Verification
 - 7.2 Performance Verification
- 8 Rollback Plan
 - 8.1 Rollback Checklist
- 9 Migration Best Practices

1 Preface

This document is the migration manual for the Kingdee Apusic Distributed Message Queue for MQTT (Apusic Distributed Message Queue, ADMQ for MQTT), detailing how to migrate from an existing MQTT Broker to ADMQ for MQTT.

1.1 Target Audience

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

1.2 Related Documents

To learn more about ADMQ for MQTT, please refer to the following ADMQ for MQTT product documentation set:

No.	Document	Description
1	ADMQ for MQTT Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for MQTT.
2	ADMQ for MQTT Installation Manual	Details how to install ADMQ for MQTT on various operating systems and how to start/stop ADMQ for MQTT services.
3	ADMQ for MQTT Message Engine User Manual	Details the usage, configuration, management, and supporting tools of ADMQ for MQTT message engine features.
4	ADMQ for MQTT Console User Manual	Details the usage and operation instructions of the ADMQ for MQTT console.
5	ADMQ for MQTT Development Manual	Details ADMQ for MQTT client application development based on various programming languages.
6	ADMQ for MQTT Migration Manual	Details the migration from MQTT Broker to ADMQ for MQTT.
7	ADMQ for MQTT Operations Manual	Details ADMQ for MQTT monitoring, operations, security hardening, and other operational tasks.
8	ADMQ for MQTT Performance Tuning Manual	Details ADMQ for MQTT performance tuning guidelines.

1.3 Technical Support

ADMQ for MQTT 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 the exception or error

2 Migration Overview

2.1 Migration Principles

ADMQ for MQTT is fully compatible with MQTT 3.1, 3.1.1, and 5.0 protocols. The migration process follows these principles:

1. **Zero code changes:** Clients do not need to modify code, only need to change the connection address
2. **Configuration compatibility:** Supports standard MQTT client and configuration parameters
3. **Data migratable:** Supports migration of users, ACL rules, retained messages, and other configuration
4. **Smooth switchover:** Supports dual-write and gray-scale switchover to reduce migration risk

2.2 Migration Scope

Migration Item	Supported	Description
Application code	Yes	No modification required, only change Broker address
Client configuration	Yes	Supports standard MQTT clients
User accounts	Yes	Supports built-in database user migration
ACL rules	Yes	Supports ACL permission rule migration
Retained messages	Yes	Supports retained message migration
Listener configuration	Yes	Supports TCP/SSL/WebSocket listener configuration
System configuration	Partial	Some advanced configurations may need manual adjustment

2.3 Pre-migration Checklist

2.3.1 Protocol Compatibility Check

Source MQTT Broker	Compatibility	Description
EMQX	Fully compatible	Recommended migration version
Mosquitto	Fully compatible	Basic MQTT features
HiveMQ	Fully compatible	Enterprise MQTT features
RabbitMQ MQTT Plugin	Fully compatible	Basic MQTT features

2.3.2 Client Compatibility Check

Client Type	Compatible Version	Description
Eclipse Paho	1.2.x+	Fully compatible
MQTT.js	4.x+	Fully compatible
EMQX Client	5.x+	Fully compatible
Other standard MQTT clients	-	Fully compatible

3 Migration Plans

3.1 Plan 1: Dual-Write Migration (Recommended)

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

Steps:

1. Deploy ADMQ for MQTT Cluster

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

2. Configure Dual-Write

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

3. Synchronize Configuration

- Create users and ACL rules in ADMQ for MQTT
- Configure listeners and system parameters

4. Switch Subscribers

- First switch some subscribers to ADMQ for MQTT
- Observe whether the business is functioning normally

5. Full Switchover

- After confirmation, switch all subscribers
- Stop sending messages to the original MQTT Broker

6. Decommission the Original Cluster

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

3.2 Plan 2: Downtime Migration

Suitable for business scenarios that can tolerate a brief downtime. Faster migration speed.

Steps:

1. Stop the Business System

- Pause all MQTT client connections
2. Export Original Broker Configuration
 - Export users, ACL rules, retained messages, etc.
 3. Migrate Configuration to ADMQ for MQTT
 - Create corresponding resources via the console or command line
 4. Switch Connection Address
 - Modify application configuration to point to ADMQ for MQTT
 5. Start the Business System
 - Restore MQTT client connections

3.3 Plan 3: Gray-Scale Migration

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

Steps:

1. Divide migration batches by business module
2. Execute each batch following 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 User Migration

4.1.1 Export Users from Original Broker

```
# EMQX example  
emqx ctl admins list  
# Or export using Dashboard
```

4.1.2 Create Users in ADMQ for MQTT

1. Log in to the console
2. Navigate to the "Users" page
3. Create a built-in database authenticator (if not yet created)
4. Batch import or create users one by one
5. Click "Sync to MQTT"

4.2 ACL Rule Migration

4.2.1 Export ACL Rules from Original Broker

```
# EMQX example  
cat etc/acl.conf
```

4.2.2 Create ACL Rules in ADMQ for MQTT

1. Log in to the console
2. Navigate to "Authentication & Authorization" - "Permissions" page
3. Select the `built_in_database` authorization source
4. Create ACL rules one by one
5. Click "Sync to MQTT"

4.3 Listener Migration

4.3.1 Export Listener Configuration from Original Broker

```
# EMQX example  
cat etc/emqx.conf | grep listener
```

4.3.2 Create Listeners in ADMQ for MQTT

1. Navigate to the "Listeners" page
2. Create TCP, SSL, WebSocket, WebSocket Secure listeners
3. SSL/WSS types require uploading certificate files

5 Retained Message Migration

5.1 Migration Using Client Scripts

For small-scale data, use client scripts to migrate retained messages:

```
import paho.mqtt.client as mqtt

# Connect to original Broker
src_client = mqtt.Client(client_id="migrate-src")
src_client.connect("old-mqtt-broker", 1883, 60)

# Connect to ADMQ for MQTT
dst_client = mqtt.Client(client_id="migrate-dst")
dst_client.connect("admq-mqtt", 1883, 60)

# Subscribe to all topics and receive retained messages
def on_message(client, userdata, msg):
    if msg.retain:
        # Forward to ADMQ for MQTT
        dst_client.publish(msg.topic, msg.payload, qos=msg.qos,
retain=True)
        print(f"Migrated retained message: {msg.topic}")

src_client.on_message = on_message
src_client.subscribe("#", qos=1)
src_client.loop_forever()
```

6 Client Migration

6.1 Connection Address Change

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

```
# Original configuration
mqtt.broker=old-mqtt-broker
mqtt.port=1883

# New configuration
mqtt.broker=admq-mqtt
mqtt.port=1883
```

6.2 SSL/TLS Configuration Change

If SSL is enabled, certificates need to be updated:

```
# Original configuration
mqtt.ssl.ca-file=/path/to/old-ca.pem

# New configuration
mqtt.ssl.ca-file=/path/to/admq-ca.pem
```

6.3 Common Client Configuration

6.3.1 Java (Spring Boot)

```
spring:
  mqtt:
    host: admq-mqtt
    port: 1883
    username: device-service
```

```
password: your-password  
client-id: spring-client-001
```

6.3.2 Python

```
client = mqtt.Client(client_id="app-001")  
client.username_pw_set("device-service", "your-password")  
client.connect("admq-mqtt", 1883, 60)
```

6.3.3 JavaScript

```
const client = mqtt.connect('mqtt://admq-mqtt:1883', {  
  clientId: 'web-client-001',  
  username: 'device-service',  
  password: 'your-password'  
});
```

7 Migration Verification

7.1 Functional Verification

Verification Item	Verification Method
Connection test	Connect to ADMQ for MQTT using a client, confirm connection is established
Message publishing	Send test messages, confirm they reach the Broker successfully
Message subscription	Start a subscriber, confirm messages are received normally
QoS 1/2	Test QoS 1/2 message delivery and acknowledgment
Retained messages	Send a retained message, confirm new subscribers receive it immediately
Will message	Simulate abnormal disconnection, confirm will message is published
ACL permissions	Test whether allow/deny rules take effect
Cluster functionality	Verify automatic failover when a node fails

7.2 Performance Verification

Verification Item	Verification Method
Concurrent connections	Test a large number of clients connecting concurrently
Message throughput	Test message publish/subscribe rate
Latency	Measure the delay from message publish to receive
Stability	Run long-duration tests, observe for any anomalies

8 Rollback Plan

If issues arise during migration, roll back to the original MQTT Broker:

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

8.1 Rollback Checklist

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

9 Migration Best Practices

1. **Thoroughly test:** Complete the full migration flow verification in a test environment
2. **Back up data:** Backup the original cluster configuration and data before migration
3. **Monitor the migration process:** Monitor various metrics in real-time during migration
4. **Create a rollback plan:** Ensure quick rollback if issues occur
5. **Choose off-peak hours:** Perform migration during business off-peak hours
6. **Gradual switchover:** First switch non-core services, verify before switching core services
7. **Keep the original cluster:** Retain the original cluster for some time after migration for query purposes

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

