



APUSIC
固若长城
睿比世界

Quick Start

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 Quick Start
 - 2.1 **Process Overview**
 - 2.2 Step 1: Create Authenticator and Authorization Source
 - 2.3 Step 2: Create User
 - 2.4 Step 3: Create Topic
 - 2.5 Step 4: Configure ACL Permissions
 - 2.6 Step 5: Send Test Message to Verify Path
 - 2.7 Step 6: Integrate Business Code
 - 2.8 Subsequent Operations
- 3 Summary

1 Preface

This document is a quick start guide for Apusic Distributed Message Queue for MQTT (Apusic Distributed Message Queue, referred to as: ADMQ for MQTT), providing a product introduction, basic steps for installation and deployment, and verification methods.

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

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

No.	Document	Description
1	Apusic Distributed Message Queue for MQTT Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for MQTT.
2	Apusic Distributed Message Queue for MQTT Installation Guide	Details how to install ADMQ for MQTT on various operating systems, as well as service start/stop operations.
3	Apusic Distributed Message Queue for MQTT Message Engine User Manual	Details the use, configuration, and management of ADMQ for MQTT message engine features and supporting tools.
4	Apusic Distributed Message Queue for MQTT Console User Manual	Details the use and operation of the ADMQ for MQTT console.
5	Apusic Distributed Message Queue for MQTT Development Guide	Details how to develop ADMQ for MQTT client applications in various development languages.
6	Apusic Distributed Message Queue for MQTT Migration Guide	Details the migration from MQTT Broker to ADMQ for MQTT.
7	Apusic Distributed Message Queue for MQTT Operations Manual	Details monitoring, operations, and security hardening for ADMQ for MQTT.
8	Apusic Distributed Message Queue for MQTT Performance Optimization Guide	Details performance tuning for ADMQ for MQTT.

1.3 Technical Support

ADMQ for MQTT provides comprehensive technical support services. You can obtain technical support through the following channels:

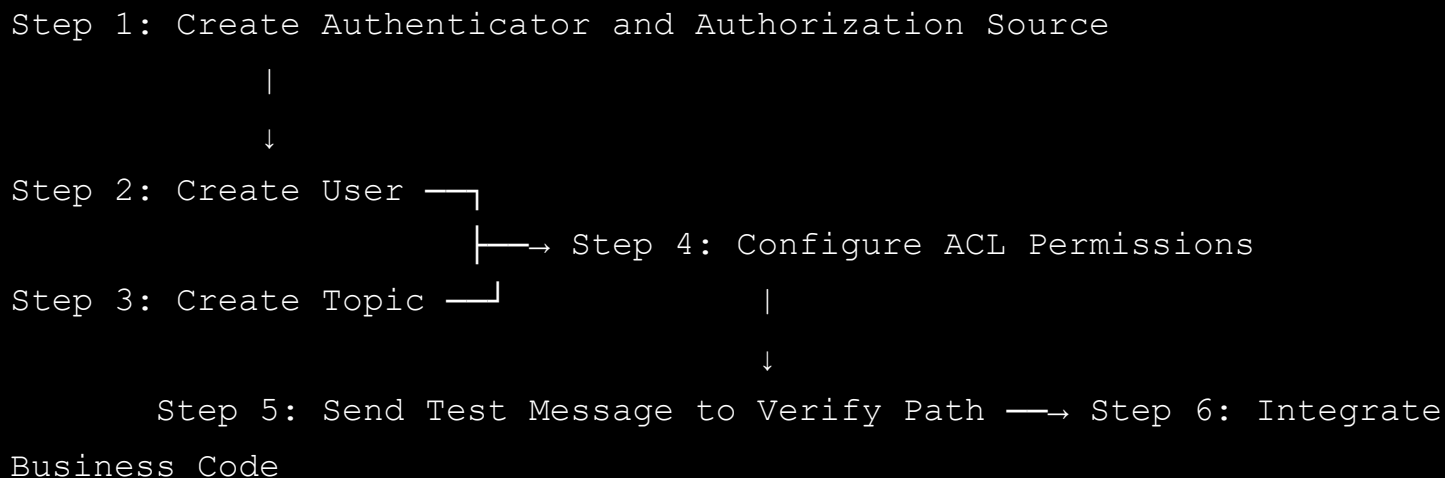
- Website: www.apusic.com
- Phone: 400-855-5800
- Email: support@apusic.com
- Kingdee Cloud Community: <https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN>

When seeking 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 Quick Start

2.1 Process Overview



2.2 Step 1: Create Authenticator and Authorization Source

MQTT users and ACL rules depend on the authenticator and authorization source respectively. When using an MQTT cluster for the first time, you need to create a built-in database authenticator and authorization source first.

1. Go to the **Authentication & Authorization** page (Left menu: Message Middleware → ADMQ MQTT → Authentication & Authorization)
2. On the **Authenticator** tab, click the "Add" button
3. Fill in:
 - Name: `built_in_database`
 - Authentication Method: `Built-in Database`
 - Description: `MQTT Built-in Database Authenticator`
4. Click "Confirm"
5. Switch to the **Authorization Source** tab, click the "Add" button
6. Fill in:
 - Name: `built_in_database`
 - Authorization Method: `Built-in Database`
 - Description: `MQTT Built-in Database Authorization Source`
7. Click "Confirm"

Why do this first? Users created later must be bound to an authenticator, and ACL rules must be bound to an authorization source. Without an authenticator and authorization source, the user and permission pages will

not have the corresponding source to select.

2.3 Step 2: Create User

Create a dedicated MQTT user for the "Device Reporting System":

1. Go to the **Users** page (Left menu: Message Middleware → ADMQ MQTT → Users)
2. Click the "Add" button
3. Fill in:
 - **Authenticator:** Select the created `built_in_database` authenticator
 - **Username:** `device-service`
 - **Password:** Set a secure password
 - **Is Super User:** `No`
 - **Description:** `Device reporting system service account`
4. Click "Confirm"

It is recommended not to enable super user for regular business accounts. Super users bypass all ACL checks.

2.4 Step 3: Create Topic

Topics are the categorization unit for messages in MQTT. Create a topic for device reporting:

1. Go to the **Topics** page (Left menu: Message Middleware → ADMQ MQTT → Topics)
2. Click the "Add" button
3. Fill in:
 - **Topic:** `sensor+/temperature`
 - **Default QoS:** `1`
 - **Retain:** `No`
 - **Description:** `Device temperature reporting topic`
4. Click "Confirm"

Topic Wildcard Explanation: `+` matches a single level, so `sensor+/temperature` can match `sensor/living-room/temperature`, but cannot match `sensor/living-room/bedroom/temperature` (the latter requires `#`).

2.5 Step 4: Configure ACL Permissions

Assign publish/subscribe topic permissions to the user:

1. Go to the **Permissions** page (Left menu: Message Middleware → ADMQ MQTT → Permissions)
2. Ensure `built_in_database` authorization source is selected at the top
3. Click the "Add" button
4. Fill in:
 - Subject Type: `Username`
 - Username: `device-service`
 - Permission: `Allow`
 - Action: `All` (allow both publish and subscribe)
 - Topic: `sensor/#`
 - QoS: `-1` (unlimited)
 - Retain: `-1` (unlimited)
 - Description: `Device service can read/write all topics under sensor`
5. Click "Confirm"

Effectiveness Note: ACL rules take effect automatically after saving, no need to click "Sync to MQTT".

2.6 Step 5: Send Test Message to Verify Path

Before integrating business code, use the platform's built-in send function to verify the message path:

1. Go to the **Topics** page
2. Click "Send Message" on the `sensor+/temperature` row
3. Fill in:
 - Payload: `{"deviceId":"D001","temperature":24.5}`
 - QoS: `1`
 - Retain: `No`
4. Click "Confirm Send"
5. Go to the **Retained Messages** page (if Retain is No, it will not appear here), or go to the **Clients** page to check if a test client is online

Verification Pass Indicator: Message sent successfully with no permission denial prompt, indicating user authentication and ACL configuration are effective.

2.7 Step 6: Integrate Business Code

Platform resources are now ready. Next, integrate the MQTT client into your application:

Connection Parameters:

Parameter	Value	How to Obtain
Broker Address	Cluster node IP	Provided by administrator
Port	1883 (TCP)	View from listener management page
Username	device-service	Created in Step 2
Password	<i>Your set password</i>	Set in Step 2
ClientID	Business-defined	Must be globally unique

Producer Example (Java + Eclipse Paho):

```
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttMessage;

MqttClient client = new MqttClient("tcp://localhost:1883",
    "publisher-001");
org.eclipse.paho.client.mqttv3.MqttConnectOptions options = new
org.eclipse.paho.client.mqttv3.MqttConnectOptions();
options.setUsername("device-service");
options.setPassword("your-password".toCharArray());
client.connect(options);

MqttMessage message = new MqttMessage("
{\"deviceId\":\"D001\", \"temperature\":24.5}").getBytes();
message.setQos(1);
client.publish("sensor/D001/temperature", message);

client.disconnect();
client.close();
```

Consumer Example (Java + Eclipse Paho):

```
MqttClient client = new MqttClient("tcp://localhost:1883",
    "subscriber-001");
org.eclipse.paho.client.mqttv3.MqttConnectOptions options = new
org.eclipse.paho.client.mqttv3.MqttConnectOptions();
```

```
options.setUsername("device-service");
options.setPassword("your-password".toCharArray());
client.connect(options);

client.subscribe("sensor+/temperature", 1);
client.setCallback(new org.eclipse.paho.client.mqttv3.MqttCallback()
{
    public void messageArrived(String topic, MqttMessage message) {
        System.out.println("Message received: " + topic + " -> " +
new String(message.getPayload()));
    }
    public void connectionLost(Throwable cause) {}
    public void
deliveryComplete(org.eclipse.paho.client.mqttv3.IMqttDeliveryToken
token) {}
});
```

2.8 Subsequent Operations

After business integration, daily operations can be managed through the following features:

Operations Scenario	Navigation Path	Description
View Online Clients	Client Management	View client ID, username, IP, connection time
Force Offline Abnormal Client	Client row → "Disconnect"	Immediately disconnect the specified connection
View Message Backlog	Topic Monitoring / Monitoring Dashboard	Monitor inflow/outflow/discard counts
Troubleshoot Consumption Delay	Slow Subscription Management	View subscriptions with long message delivery time
Manage Retained Messages	Retained Message Management	View or delete unwanted retained messages
View System Alerts	Monitoring & Statistics → Alerts	View alerts for CPU/memory/connection count anomalies

Adjust Authentication/Authorization	Authentication & Authorization	Modify authenticators, authorization sources, ACL rules
--	-----------------------------------	--

3 Summary

After completing the above 6 steps, you have:

-  Created MQTT built-in database authenticator and authorization source
-  Created a dedicated MQTT business user
-  Created a device reporting topic
-  Configured fine-grained ACL access control
-  Verified the correctness of the message path
-  Integrated configuration parameters into business code

This is the **minimum viable closed loop** for using MQTT. Based on business needs, you can further extend with more topics, users, ACL rules, or configure listeners and system parameters for richer functionality.

For detailed descriptions of each feature, please refer to other documents in the same directory.

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

