



Development Manual

金蝶Apusic分布式消息队列V2.0.6_for_kafka

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 Core Architecture
 - 2.1 Topics and Partitions
 - 2.2 Replication Mechanism
 - 2.3 High Availability and Failover
- 3 Installation and Deployment
 - 3.1 Standalone Deployment
- 4 Client Development
 - 4.1 Maven Dependency
 - 4.2 Producer Core Configuration
 - 4.2.1 Producer Example Code
 - 4.3 Consumer Core Configuration
 - 4.3.1 Consumer Example Code
- 5 Production Best Practices
 - 5.1 Topic Design
 - 5.2 Performance Tuning
 - 5.3 Monitoring Metrics
- 6 Security Configuration
 - 6.1 SASL/PLAIN Authentication Configuration
 - 6.2 ACL Example
- 7 Common Commands
 - 7.1 Topic Management
 - 7.2 Consumer Group Management
 - 7.3 KRaft Mode Operations
- 8 Troubleshooting
 - 8.1 Common Issues
 - 8.2 Log Locations

1 Preface

This document provides development instructions for the Kingdee Apusic Distributed Message Queue for Kafka (Apusic Distributed Message Queue, hereinafter referred to as ADMQ for Kafka), helping users quickly learn how to develop using ADMQ for Kafka.

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

For more information about ADMQ for Kafka, please refer to the following ADMQ for Kafka documentation set:

No.	Document	Description
1	ADMQ for Kafka Quick Start Guide	Briefly introduces how to quickly get started with ADMQ for Kafka.
2	ADMQ for Kafka Installation Guide	Details how to install ADMQ for Kafka on various operating systems, as well as service start/stop operations.
3	ADMQ for Kafka Message Engine User Manual	Details the usage, configuration, management, and accompanying tools of the ADMQ for Kafka message engine.
4	ADMQ for Kafka Console User Manual	Details the functional usage and operation instructions of the ADMQ for Kafka console.
5	ADMQ for Kafka Development Guide	Details client application development for ADMQ for Kafka using various programming languages.
6	ADMQ for Kafka Migration Guide	Details the process of migrating from Kafka to ADMQ for Kafka.
7	ADMQ for Kafka Operations Guide	Details monitoring, operations, security hardening, and other operational instructions for ADMQ for Kafka.
8	ADMQ for Kafka Performance Optimization Guide	Details performance tuning for ADMQ for Kafka.

1.3 Technical Support

ADMQ for Kafka 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 requesting 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 logs, screenshots, etc. of the exception or error

2 Core Architecture

2.1 Topics and Partitions

- **Topic:** Logical classification of messages
- **Partition:** Physical shard of a Topic, enabling parallel reading/writing and horizontal scaling
 - Messages within a single partition are strictly ordered
 - No global order guarantee across different partitions
 - The number of partitions should be set based on throughput requirements, typically 10-50

2.2 Replication Mechanism

- **Leader Replica:** Handles all read and write requests
- **Follower Replica:** Only syncs data from the Leader, does not handle client requests
- **ISR (In-Sync Replicas):** The set of replicas that are in sync with the Leader
 - Only replicas in the ISR are eligible to become a new Leader
 - Controlled by `replica.lag.time.max.ms` (default 30 seconds)

2.3 High Availability and Failover

- Controller monitors Broker status
- When a Leader fails, a new Leader is elected from the ISR (the replica with the highest offset is preferred)
- Cluster metadata is updated, and clients automatically reconnect to the new Leader
- Non-ISR replicas are prohibited from becoming Leader by default
(`unclean.leader.election.enable=false`)

3 Installation and Deployment

3.1 Standalone Deployment

Extract the installation package

```
mkdir /apusic  
tar -zxf ADMQ-V2.0.6.391-Kafka-20260319.tar.gz -C /apusic
```

Modify configuration

```
cd /apusic/admq-kafka  
vi config/kafka-standalone.conf  
...  
# Change the address below to the actual server IP  
listeners=PLAINTEXT://192.168.1.10:9092  
...
```

Start

```
bin/admq-daemon start kafka zk  
bin/admq-daemon start kafka standalone
```

4 Client Development

4.1 Maven Dependency

```
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>3.9.1</version>
</dependency>
```

4.2 Producer Core Configuration

Config	Recommended Value	Description
bootstrap.servers	host1:9092,host2:9092	Cluster address list
acks	all	Wait for acknowledgement from all ISR replicas
enable.idempotence	true	Enable idempotence to prevent duplicates
compression.type	lz4 Or zstd	Compression algorithm
linger.ms	5-10	Batch send delay
batch.size	16384	Batch size (bytes)
retries	Integer.MAX_VALUE	Number of retries
max.in.flight.requests.per.connection	5	Max unacknowledged requests per connection (≤ 5 when idempotence enabled)

4.2.1 Producer Example Code

```
Properties props = new Properties();
props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092");
```

```

props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
StringSerializer.class.getName());
props.put(ProducerConfig.ACKS_CONFIG, "all");
props.put(ProducerConfig.ENABLE_IDEMPOTENCE_CONFIG, "true");
props.put(ProducerConfig.RETRIES_CONFIG, Integer.MAX_VALUE);
props.put(ProducerConfig.COMPRESSION_TYPE_CONFIG, "lz4");

KafkaProducer<String, String> producer = new KafkaProducer<>(props);

ProducerRecord<String, String> record = new ProducerRecord<>("my-
topic", "key1", "value1");
producer.send(record, (metadata, exception) -> {
    if (exception != null) {
        exception.printStackTrace();
    } else {
        System.out.println("Sent to partition " +
metadata.partition() +
        ", offset " + metadata.offset());
    }
});

producer.close();

```

4.3 Consumer Core Configuration

Config	Recommended Value	Description
bootstrap.servers	host1:9092,host2:9092	Cluster address
group.id	Custom	Consumer group ID
enable.auto.commit	false	Disable auto-commit, manual control
auto.offset.reset	earliest Or latest	Strategy when no offset exists
max.poll.records	500	Max records per single poll

session.timeout.ms	30000	Session timeout
heartbeat.interval.ms	10000	Heartbeat interval (should be less than 1/3 of session timeout)

4.3.1 Consumer Example Code

```

Properties props = new Properties();
props.put(ConsumerConfig.BOOTSTRAP_SERVERS_CONFIG,
"localhost:9092");
props.put(ConsumerConfig.GROUP_ID_CONFIG, "my-group");
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
StringDeserializer.class.getName());
props.put(ConsumerConfig.ENABLE_AUTO_COMMIT_CONFIG, "false");
props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");

KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
consumer.subscribe(Collections.singletonList("my-topic"));

try {
    while (true) {
        ConsumerRecords<String, String> records =
consumer.poll(Duration.ofMillis(100));
        for (ConsumerRecord<String, String> record : records) {
            System.out.println("Received: " + record.value());
        }
        consumer.commitSync();
    }
} finally {
    consumer.close();
}

```

5 Production Best Practices

5.1 Topic Design

- Partition count calculation: Single partition throughput $\times$ number of partitions $\geq$ target throughput
- Replication factor: 3 replicas recommended for production
- Cleanup policy:
 - `cleanup.policy=delete` : Clean by time/size
 - `cleanup.policy=compact` : Log compaction, retain the latest Key

5.2 Performance Tuning

- Producer side:
 - Enable compression (`lz4` or `zstd`)
 - Set `linger.ms` and `batch.size` appropriately
 - Enable idempotence and transactions (Exactly-Once semantics)
- Consumer side:
 - Adjust `max.poll.records` to avoid slow single processing
 - Set `fetch.min.bytes` and `fetch.max.wait.ms` appropriately

5.3 Monitoring Metrics

Metric	Description	Alert Threshold
<code>UnderReplicatedPartitions</code>	Number of under-replicated partitions	> 0
<code>OfflinePartitionsCount</code>	Number of offline partitions	> 0
<code>RequestLatencyAvg</code>	Average request latency	$> 100\text{ms}$
<code>MessagesInPerSec</code>	Messages per second	Based on business
<code>BytesInPerSec</code>	Bytes written per second	Based on business

6 Security Configuration

6.1 SASL/PLAIN Authentication Configuration

```
# broker configuration
listeners=SASL_PLAINTEXT://0.0.0.0:9092
security.inter.broker.protocol=SASL_PLAINTEXT
sasl.mechanism.inter.broker.protocol=PLAIN
sasl.enabled.mechanisms=PLAIN

authorizer.class.name=kafka.security.authorizer.AclAuthorizer
allow.everyone.if.no.acl.found=false

# JAAS configuration file (passwords below are examples only; use
strong passwords in production)
KafkaServer {
    org.apache.kafka.common.security.plain.PlainLoginModule required
    username="admin"
    password="admin-secret"
    user_admin="admin-secret"
    user_alice="alice-secret";
};
```

6.2 ACL Example

```
# Grant alice read and write permissions on topic my-topic
kafka-acls.sh --bootstrap-server localhost:9092 \
  --add --allow-principal User:alice \
  --operation Read --operation Write \
  --topic my-topic

# Grant alice operation permission on consumer group my-group
kafka-acls.sh --bootstrap-server localhost:9092 \
```

```
--add --allow-principal User:alice \  
--operation Read --group my-group
```

7 Common Commands

ADMQ for Kafka provides native commands located in the `admq-kafka/kafka/bin/` directory.

7.1 Topic Management

```
cd /apusic/admq-kafka/kafka/bin/

# Create topic
kafka-topics.sh --bootstrap-server localhost:9092 --create \
  --topic my-topic --partitions 3 --replication-factor 3

# List all topics
kafka-topics.sh --bootstrap-server localhost:9092 --list

# View topic details
kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic
my-topic

# Modify partition count (can only increase)
kafka-topics.sh --bootstrap-server localhost:9092 --alter \
  --topic my-topic --partitions 6

# Delete topic
kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic
my-topic
```

7.2 Consumer Group Management

```
# List consumer groups
kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list

# View consumer group details
kafka-consumer-groups.sh --bootstrap-server localhost:9092 \
```

```
--describe --group my-group

# Reset offsets
kafka-consumer-groups.sh --bootstrap-server localhost:9092 \
  --group my-group --reset-offsets --to-earliest \
  --topic my-topic --execute
```

7.3 KRaft Mode Operations

```
# Format storage directory (first time only)
kafka-storage.sh format -t <cluster-id> -c
config/kraft/server.properties

# View quorum status
kafka-metadata-quorum.sh --bootstrap-server localhost:9092 describe
--status

# Dynamically add controller node
kafka-metadata-quorum.sh --bootstrap-server localhost:9092 add \
  --controller-quorum-voter <node-id>@<host>:<port>
```

8 Troubleshooting

8.1 Common Issues

Issue	Possible Cause	Solution
Frequent consumer rebalancing	<code>session.timeout.ms</code> too small or processing time too long	Increase session timeout, optimize consumption logic
Message loss	<code>acks=1</code> or idempotence not enabled	Set <code>acks=all</code> , enable <code>enable.idempotence</code>
Duplicate consumption	Offset commit failure	Use manual commit, handle deduplication logic
High producer latency	Network or disk bottleneck	Check network bandwidth, optimize disk I/O
Partition unavailable	Broker failure or insufficient replicas	Check Broker status, ensure sufficient ISR

8.2 Log Locations

- Broker logs: `logs/server.log`
- Controller logs: `logs/controller.log`
- State change logs: `logs/state-change.log`

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

