Apusic 金蝶天燕

生成时间：2026-07-13

消息引擎用户手册

金蝶Apusic分布式消息队列V2.0.6_for_rocketmq

版权声明

本文档所涉及的软件著作权、版权等知识产权已依法进行了注册，由金蝶天燕云计算股份有限公司合法拥有。受《中华人民共和国著作权法》《计算机软件保护条例》《知识产权保护条例》和相关国际版权条约、法律、法规以及其它知识产权法律和条约的保护。未经授权许可，不得非法使用。

免责声明

本文档包含的版权信息由金蝶天燕云计算股份有限公司合法拥有，受法律的保护，金蝶天燕云计算股份有限公司对本文档可能涉及到的非金蝶天燕云计算股份有限公司的信息不承担任何责任。在法律允许的范围内，您可以查阅并仅能够在《中华人民共和国著作权法》规定的合法范围内复制和打印本文档。任何单位和个人未经金蝶天燕云计算股份有限公司书面授权许可，不得使用、修改、再发布本文档的任何部分和内容，否则将被视为侵权，金蝶天燕云计算股份有限公司有依法追究其责任的权利。

本文档如有更新，不另行通知。对本文档中的问题您可向金蝶天燕云计算股份有限公司告知或查询。未经本公司明确授予的任何权利均予保留。

商标声明

Apusic是深圳市金蝶天燕云计算股份有限公司向中华人民共和国国家商标局申请注册的注册商标，注册商标专用权由金蝶天燕合法拥有，受法律保护。未经金蝶天燕的书面许可，任何单位及个人不得以任何方式或理由对该商标的任何部分进行使用、复制、修改、传播、抄录或与其它产品捆绑使用销售。凡侵犯金蝶天燕商标权的，金蝶天燕将依法追究其法律责任。本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

目录

1 前言

本文档为金蝶Apusic分布式消息队列for RocketMQ（Apusic Distributed Message Queue for RocketMQ，简称：ADMQ for RocketMQ）消息引擎的用户手册，详细介绍了ADMQ for RocketMQ消息引擎的功能使用、配置方法及管理操作等内容。

1.1 适用对象

本文档适用于IT信息化业务负责人、研发经理、软件项目经理、软件架构师、运维工程师。

1.2 相关文档

了解更多ADMQ for RocketMQ产品相关的信息，请参阅以下ADMQ for RocketMQ产品手册文档集：

序号	手册文档	说明
1	金蝶Apusic分布式消息队列for RocketMQ 快速使用手册	简单介绍了如何快速上手使用ADMQ for RocketMQ 。
2	金蝶Apusic分布式消息队列for RocketMQ 安装手册	详细介绍如何在各操作系统上安装ADMQ for RocketMQ，以及ADMQ for RocketMQ服务启停等操作。
3	金蝶Apusic分布式消息队列for RocketMQ 消息引擎用户手册	详细介绍 ADMQ for RocketMQ 消息引擎相关功能的使用、配置、管理及配套工具的使用方法。
4	金蝶Apusic分布式消息队列for RocketMQ 管控台用户手册	详细介绍ADMQ for RocketMQ管控台相关功能的使用和操作说明。
5	金蝶Apusic分布式消息队列for RocketMQ 开发手册	详细介绍基于各开发语言进行ADMQ for RocketMQ客户端应用开发的说明。
6	金蝶Apusic分布式消息队列for RocketMQ 迁移手册	详细介绍从RocketMQ迁移到ADMQ for RocketMQ的说明。
7	金蝶Apusic分布式消息队列for RocketMQ 运维手册	详细介绍ADMQ for RocketMQ的监控、运维、安全加固等运维说明。
8	金蝶Apusic分布式消息队列for RocketMQ 性能优化手册	详细介绍ADMQ for RocketMQ性能调优的说明。

1.3 技术支持

ADMQ for RocketMQ产品提供全面的技术支持服务，您可以通过以下方式获得技术支持：

- 网址：www.apusic.com
- 电话：400-855-5800
- 邮箱：support@apusic.com
- 金蝶云社区：https://vip.kingdee.com/?productId=73&productLineId=14&lang=zh-CN

您在取得技术支持时，请提供如下信息：

1. 您的姓名

2. 公司信息与联系方式

3. 操作系统及其版本

4. 产品版本号

5. 出现异常及错误的日志、截图等详细信息

2 核心引擎目录说明

目录名	说明
`bin`	存放启动、停止及管理脚本
`conf`	存放配置文件 (如 `broker.conf,nameserver.properties`)
`lib`	存放 RoekctMQ 依赖包
`logs`	存放运行日志 (Broker 日志、NameServer 日志等)

3 Nameserver 配置说明

3.1 基础配置（Basic Configuration）

参数	说明	生产环境建议
listenPort	Nameserver的监听端口	默认使用9876
deployServerAddress	Nameserver的地址配置参数	默认使用127.0.0.1，可以更新成对应机器地址
clusterName	集群名称	默认defaultCluster
managerUrl	管控台的接入地址	根据实际部署的地址IP填写，如果没有管控台，默认参数可不修改

3.2 TLS配置

核心作用：tls认证加密

参数	说明	生产环境建议
tls.enable	TLS认证开启参数	默认false
tls.server.keyPath	存放服务器证书密钥的地址	根据实际存放文件地址填写
tls.server.keyPassword	服务器证书密钥密码	根据实际设置的密码填写
tls.server.certPath	存放服务器证书的地址	根据实际存放文件地址填写
tls.server.trustCertPath	存放服务器信任证书的地址	根据实际存放文件地址填写
tls.client.keyPath	存放客户端密钥的地址	根据实际存放文件地址填写
tls.client.keyPassword	客户端证书密钥密码	根据实际设置的密码填写
tls.client.certPath	存放客户端证书的地址	根据实际存放文件地址填写
tls.client.trustCertPath	存放客户端信任证书的地址	根据实际存放文件地址填写

4 Broker 配置说明

4.1 基础配置（Basic Configuration）

参数	说明	生产环境建议
brokerClusterName	集群名称	默认集群名称defaultCluster
brokerName	broker节点名称	默认节点名称broker-a
brokerRole	broker角色	默认ASYNC_MASTER
brokerId	broker id	根据实际部署的broker设置
brokerIP1	broker 映射IP	broker映射的IP
listenPort	监听端口	默认端口10911
haListenPort	主从同步的监听端口	默认端口9888
managerUrl	管控台的接入地址	根据实际部署的地址IP填写，如果没有管控台，默认参数可不修改

4.2 NameServer相关配置

参数	说明	生产环境建议
namesrvAddr	nameserver连接地址	nameserver组件连接地址

4.3 存储配置

核心作用：存储管理

参数	说明	生产环境建议
storePathRootDir	存储路径	根据实际使用设置
storePathCommitLog	commitlog存储路径	根据实际使用设置
storePathConsumerQueue	消费队列存储路径	根据实际使用设置
storePathIndex	index存储路径	根据实际使用设置
log.dir	日志存储地址	根据实际使用设置

4.4 Promethus配置

核心作用：监控运维

参数	说明	生产环境建议
enableMultiDispatch	是否开启多维度消息分发的参数	建议开启，设置成true

enablemq	是否开启多维度消息分发的参数	建议开启，设置成true
metricsPromExporterHost	promethus监控地址	默认0.0.0.0
metricsExporterType	监控类型	默认PROM

5 核心概念

5.1 RocketMQ 架构

ADMQ for RocketMQ 完全兼容 RocketMQ 核心架构，主要包含以下组件：

组件	说明
NameServer	轻量级服务注册与发现中心，维护 Broker 路由信息
Broker	消息存储和转发服务节点，分 Master 和 Slave
Producer	生产者，负责发送消息到 Broker
Consumer	消费者，负责从 Broker 接收消息
Topic	消息主题，消息的分类标识
MessageQueue	消息队列，Topic 的分片单位

5.2 消息模型

```
Producer -> Topic -> MessageQueue -> Consumer
```

1. 生产者发送消息到指定 Topic
2. Topic 被划分为多个 MessageQueue，分布在不同 Broker 上
3. 消费者订阅 Topic，从 MessageQueue 拉取消息消费
4. 消费进度以 Offset 形式记录

6 常见命令

6.1 Topic 管理

6.1.1 Topic 属性

属性	说明
读写队列数	每个 Broker 上的队列数量
权限	读写权限控制
顺序类型	是否顺序消息
消息类型	普通/事务/定时消息

6.1.2 创建 Topic

创建一个名为 `test-topic` 的主题，包含8个读队列，8个写队列

```
# 使用 mqadmin 创建 Topic
bin/mqadmin updateTopic -n localhost:9876 -c DefaultCluster -t test-topic -r 8 -w
8

# 参数说明
# -n: NameServer 地址
# -c: 集群名称
# -t: Topic 名称
# -r: 读队列数
# -w: 写队列数
```

6.1.3 查看 Topic

```
# 查看 Topic 列表
bin/mqadmin topicList -n localhost:9876

# 查看 Topic 路由信息
bin/mqadmin topicRoute -n localhost:9876 -t test-topic

# 查看 Topic 状态
bin/mqadmin topicStatus -n localhost:9876 -t test-topic
```

6.1.4 删除 Topic

```
bin/mqadmin deleteTopic -n localhost:9876 -c DefaultCluster -t test-topic
```

6.2 Group 管理

6.2.1 创建 Group

```
bin/mqadmin  updateSubGroup -n localhost:9876 -b localhost:10911 -g groupName
```

6.2.2 查看 Group

查看所有订阅组列表及进度

```
bin/mqadmin consumerProgress -n localhost:9876
```

查看订阅组的连接

```
bin/mqadmin consumerConnection -n localhost:9876 -g groupName
```

6.2.3 删除 Group

```
bin/mqadmin deleteSubGroup -n localhost:9876 -b localhost:10911 -g groupName
```

6.3 消息查询

根据key查询消息

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

根据消息id查询消息

```
bin/mqadmin queryMsgById -n localhost:9876 -i message-id
```

6.4 配置管理

获取broker配置

```
# 配置管理
bin/mqadmin getBrokerConfig -n localhost:9876 -b broker-a:10911
```

更新broker配置

```
bin/mqadmin updateBrokerConfig -n localhost:9876 -b broker-a:10911 -k key -v
value
```

6.5 消息轨迹

```
# 查看消息轨迹
bin/mqadmin printMsg -n localhost:9876 -t test-topic -s "2026-01-01#00:00:00:000"
-e "2026-01-02#00:00:00:000"
```

6.6 MessageQueue

MessageQueue 是 Topic 的逻辑分片，每个 Topic 在 Broker 上划分为多个队列。

6.6.1 队列分配策略

策略	说明
轮询	消息依次发送到各队列
哈希	按 Key 哈希选择队列
顺序	按指定规则选择队列

7 生产者（Producer）

7.1 发送方式

发送方式	说明	适用场景
同步发送	发送后等待服务端响应	需要可靠性的场景
异步发送	发送后立即返回，通过回调获取结果	需要高吞吐的场景
单向发送	发送后不等待响应	可靠性要求低的场景
批量发送	批量发送多条消息	需要高吞吐的场景
事务发送	两阶段提交发送	需要事务一致性的场景

7.2 消息属性

属性	说明
Topic	消息主题
Tag	消息标签，用于过滤
Key	业务唯一标识
Body	消息体
Properties	扩展属性

7.3 消息路由

生产者发送消息时，根据以下规则选择 MessageQueue：

1. **未指定 Key**：轮询选择队列
2. **指定 Key**：按 Key 哈希选择队列（同一 Key 的消息进入同一队列）
3. **顺序消息**：按消息队列选择器选择队列

8 消费者（Consumer）

8.1 消费模式

8.1.1 集群消费（Clustering）

- 同一消费者组内消息仅被消费一次
- 消息被组内某个消费者消费
- 适用场景：大部分业务场景

8.1.2 广播消费（Broadcasting）

- 同一消费者组内每条消息都被所有消费者消费
- 适用场景：本地缓存更新、配置同步

8.2 消费方式

消费方式	说明
推模式（Push）	Broker 主动推送消息到消费者
拉模式（Pull）	消费者主动拉取消息

8.3 消费位点（Offset）

8.3.1 位点类型

类型	说明
集群位点	消费进度存储在 Broker 上，集群共享
广播位点	消费进度存储在消费者本地，各自维护

8.3.2 位点管理

```
# 查看消费者组消费进度
bin/mqadmin consumerProgress -n localhost:9876 -g test-group

# 重置消费位点
bin/mqadmin resetOffsetByTime -n localhost:9876 -g test-group -t test-topic -s
"2026-01-01#00:00:00:000"
```

8.4 消息过滤

8.4.1 Tag 过滤

消费者订阅时指定 Tag，只接收匹配的消息：

```
// 订阅所有 Tag
consumer.subscribe("test-topic", "*");

// 订阅指定 Tag
consumer.subscribe("test-topic", "tag-a");

// 订阅多个 Tag（用 || 分隔）
consumer.subscribe("test-topic", "tag-a || tag-b");
```

8.4.2 SQL92 过滤

使用 SQL92 语法进行复杂过滤：

```
// 订阅满足 SQL 条件的消息
consumer.subscribe("test-topic", MessageSelector.bySql("age > 18 AND city =
'Beijing'"));
```

9 高级消息特性

9.1 顺序消息

9.1.1 全局顺序

所有消息按发送顺序消费，性能较低。

9.1.2 分区顺序

同一分区（MessageQueue）内的消息按顺序消费，不同分区间无序。

```
// 发送顺序消息
Message msg = new Message("order-topic", "tag-a", "order-1",
"content".getBytes());
producer.send(msg, new MessageQueueSelector() {
    @Override
    public MessageQueue select(List<MessageQueue> mqs, Message msg, Object arg) {
        // 按订单 ID 选择队列
        Long orderId = (Long) arg;
        long index = orderId % mqs.size();
        return mqs.get((int) index);
    }
}, orderId);

// 消费顺序消息
consumer.registerMessageListener(new MessageListenerOrderly() {
    @Override
    public ConsumeOrderlyStatus consumeMessage(List<MessageExt> msgs,
ConsumeOrderlyContext context) {
        // 按顺序消费
        return ConsumeOrderlyStatus.SUCCESS;
    }
});
```

9.2 事务消息

基于两阶段提交实现分布式事务：

```
// 创建事务生产者
TransactionMQProducer producer = new TransactionMQProducer("tx-group");
producer.setTransactionListener(new TransactionListener() {
    @Override
```

```
    public LocalTransactionState executeLocalTransaction(Message msg, Object arg)
{
        // 执行本地事务
        try {
            // 执行业务逻辑
            return LocalTransactionState.COMMIT_MESSAGE;
        } catch (Exception e) {
            return LocalTransactionState.ROLLBACK_MESSAGE;
        }
    }

    @Override
    public LocalTransactionState checkLocalTransaction(MessageExt msg) {
        // 回查本地事务状态
        if (isTransactionSuccess(msg)) {
            return LocalTransactionState.COMMIT_MESSAGE;
        }
        return LocalTransactionState.ROLLBACK_MESSAGE;
    }
});
```

9.3 定时/延时消息

支持指定时间投递消息：

```
// 发送延时消息（延时级别）
Message msg = new Message("delay-topic", "tag-a", "content".getBytes());
// 设置延时级别（1s 5s 10s 30s 1m 2m 3m 4m 5m 6m 7m 8m 9m 10m 20m 30m 1h 2h）
msg.setDelayTimeLevel(3);  // 延时 10 秒
producer.send(msg);

// RocketMQ 5.x 支持任意时间
msg.setDelayTimeMs(3600000);  // 延时 1 小时
```

9.4 批量消息

批量发送消息提高吞吐：

```
List<Message> messages = new ArrayList<>();
for (int i = 0; i < 100; i++) {
    messages.add(new Message("batch-topic", "tag-a", ("msg-" + i).getBytes()));
```

```
}
producer.send(messages);
```

10 集群与高可用

10.1 集群状态

```
# 查看 Broker 状态
bin/mqadmin brokerStatus -n localhost:9876 -b broker-a:10911
```

10.2 集群运维命令

```
# 查看集群状态
bin/mqadmin clusterList -n localhost:9876

# 查看 Broker 状态
bin/mqadmin brokerStatus -n localhost:9876 -b broker-a:10911

# 查看 Topic 列表
bin/mqadmin topicList -n localhost:9876

# 查看消费者组连接
bin/mqadmin consumerConnection -n localhost:9876 -g test-group

# 查看生产者连接
bin/mqadmin producerConnection -n localhost:9876 -t test-topic -g test-group
```

11 安全配置

11.1 ACL（1.0-适用4.x版本） 配置

11.1.1 配置文件修改

```
# conf/plain_acl.yml
globalWhiteRemoteAddresses:
  - 10.*.*.*
  - 192.168.*.*

accounts:
  - accessKey: test-user
    secretKey: test-password
    whiteRemoteAddress:
    admin: false
    defaultTopicPerm: DENY
    defaultGroupPerm: SUB
    topicPerms:
      - test-topic=CREATE+WRITE+READ
    groupPerms:
      - test-group=READ
```

11.1.2 启用 ACL

```
# 启用 ACL 插件
bin/mqadmin updateBrokerConfig -n localhost:9876 -b broker-a:10911 -k aclEnable -
v true
```

11.2 ACL （2.0-适用5.x版本）配置

11.2.1 配置文件修改

修改broker.conf

```
# 启用认证
authenticationEnabled = true
authenticationMetadataProvider =
org.apache.rocketmq.auth.authentication.provider.LocalAuthenticationMetadataProvider

# 启用授权
```

```
authorizationEnabled = true
authorizationMetadataProvider =
org.apache.rocketmq.auth.authorization.provider.LocalAuthorizationMetadataProvider

# 初始化管理员用户（首次启动自动创建）
initAuthenticationUser = {"username":"rocketmq","password":"12345678"}

# 组件间认证凭证（用于Broker主从同步、集群内部通信等）
innerClientAuthenticationCredentials =
{"accessKey":"rocketmq","secretKey":"12345678"}
```

通过修改broker.conf，需要重启集群

11.2.2 配置mqadmin工具

启用ACL后，需要配置mqadmin工具的认证凭证才能执行管理命令。

编辑 `conf/tools.yml` 文件：

```
# 使用初始化的管理员用户凭证
accessKey: rocketmq
secretKey: 12345678
```

11.2.3 创建业务用户和授权

生产者

```
# 创建生产者用户
bin/mqadmin createUser -n 127.0.0.1:9876 -c DefaultCluster \
  -u producer_user \
  -p producer123 \
  -t Normal

# 授予Topic发送权限
bin/mqadmin createAcl -n 127.0.0.1:9876 -c DefaultCluster \
  -s User:producer_user \
  -r Topic:TestTopic \
  -a Pub \
  -d Allow
```

消费者

```
# 创建消费者用户
bin/mqadmin createUser -n 127.0.0.1:9876 -c DefaultCluster \
```

```
  -u consumer_user -p consumer123 -t Normal

# 授予消费者消费权限
bin/mqadmin createAcl -n 127.0.0.1:9876 -c DefaultCluster \
  -s User:consumer_user -r Topic:TestTopic,Group:TestGroup -a Sub -d Allow
```

11.2.4 验证配置

使用Java客户端发送一条测试消息：

```
SessionCredentials credentials = new SessionCredentials("producer_user",
"producer123");
StaticSessionCredentialsProvider credentialsProvider =
    new StaticSessionCredentialsProvider(credentials);

ClientConfiguration clientConfiguration = ClientConfiguration.newBuilder()
    .setEndpoints("127.0.0.1:10911")
    .setCredentialProvider(credentialsProvider)
    .build();
// ... 创建Producer并发送消息
```

11.3 TLS 配置

```
# broker.conf
tls.server.mode=enforcing
tls.server.certPath=/path/to/server.crt
tls.server.keyPath=/path/to/server.key
tls.server.trustCertPath=/path/to/ca.crt

tls.client.keyPath=conf/auth/client.key
tls.client.keyPassword=123456
tls.client.certPath=conf/auth/client.pem
tls.client.trustCertPath=conf/auth/ca.pem
```

全国统一服务热线

4008-555-800

金蝶天燕云计算股份有限公司(简称“金蝶天燕云”)成立于2000年，前身为“金蝶中间件公司”，是金蝶集团旗下新一代软件基础云平台服务商，云计算国家标准制定企业，国家信创产业核心软件企业。金蝶天燕是国家863重点研发计划与核高基重大专项承接企业，也是“两网一站四库十二金”国家重点工程的基础平台提供商，产品广泛应用于政府、军工、金融、能源等关键行业，累计服务客户总数超过10万家。

Apusic 金蝶天燕

云计算国家标准制定企业

金蝶集团旗下基础软件企业

信息技术应用创新核心企业

官网：www.apusic.com