消息队列 CMQ 版

快速入门

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

快速入门
队列模型快速入门

最近更新时间：2024-08-19 10:21:01

操作场景

本文为您介绍从零开始创建一个队列服务并使用 Java SDK 进行收发消息测试的方法，帮助您快速了解客户端接入 TDMQ CMQ 版所需的基本操作。

前提条件

已 注册腾讯云账号 。

操作步骤

步骤1：创建队列服务

1. 登录 TDMQ CMQ 版控制台 。
2. 在左侧导航栏选择**队列服务**，选择地域后，单击**新建**，配置队列服务相关属性值。


新建队列
队列名称 *
请输入队列名称、创建后不能修改
不能为空，以字母起始，只能包含字母、数字、“-”及“_”，最大64字符，不区分大小写
资源标签
标签键
标签值
添加
配置队列属性
消息最长未确认时间
秒
范围在30秒到12小时
消息接收长轮询等待时间
0
秒
范围在0秒到30秒，推荐设置为3秒。
取出消息隐藏时长
30
秒
范围在1秒到12小时
不可见消息数量上限
100000
消息堆积容量上限
10GB
死信队列设置
死信队列
堆积消息和回溯设置
消息回溯
可回溯时间范围
1
天
1~15天，设置较长可能会产生昂贵的存储费用
可回溯存储空间
1
GB
1~10GB，设置较大可能会产生昂贵的存储费用
提交
关闭

属性	说明	取值

队列名称	QueueName，为队列的名称。	作为资源的唯一标识，调用 API 接口进行操作时，以 Queue name 为准，创建成功后无法修改。不区分大小写，相同字母即会判定为同名，并不能以 -retry 和 -dlq 结尾。
资源标签	选填，标签可以帮助您从各种维度方便地对 TDMQ CMQ 版资源进行分类管理，具体使用方法可参见 标签管理 。	-
消息最长未确认时间	如果消费客户端在获取到消息后超过此时间仍未进行消息的确认，则服务端会自动确认该消息。	范围在30秒到12小时
消息接收长轮询等待时间	PollingWaitSeconds，长轮询等待时，一个消息消费请求只会在取到有效消息或长轮询超时时才返回响应，类似于 Ajax 请求的长轮询。	范围在0秒到30秒，推荐设置为3秒，设置过高可能造成消息重复的概率提升。
取出消息隐藏时长	该项为队列的 VisibilityTimeout 属性，每条 Message 都有个默认的VisibilityTimeout，Worker 在接收到消息后，timeout 就开始计时了。如果 Worker 在 timeout 时间内没能处理完 Message，那么消息就有可能被其他 Worker 接收到并处理。	范围在1秒到12小时
不可见消息数	消息被消费者取走后，转换为不可见（Inactive），如果在经过了 隐藏时间（VisibilityTimeout）后仍未被消费，则会重新转换为可见（Active）。需要消费者在线时才能统计数据，无消费者在线时展示为 0。 不可见消息过多一般是客户端未及时 ACK 导致的，产生不可见消息会消耗一定的内存，因此为了保证队列的稳定性，消费完消息后需要尽快 ACK。	
可见消息数	普通消息（非延时消息）被发送到普通消息队列时， 初始状态为可见（Active），该状态下消息可以被消费者消费。此处显示的数据刷新会有分钟级的延迟，需要消费者在线时才能统计数据，无消费者在线时展示为 0。	
消息堆积容量上限	消息堆积一般是生产速率大于消费速率或者消费出现阻塞导致的，产生堆积会消耗一定的磁盘存储，因此为每个队列设置了一定的容量上限。	10GB，如需提升额度，请联系技术支持。

死信队列	死信队列用于处理无法被正常消费的消息。达到最大重试次数后，若消费依然失败，则表明消费者在正常情况下无法正确地消费该消息，此时，MQ 不会立刻将消息丢弃，而是将其发送到该消费者对应的特殊队列中。	-
消息回溯	若未开启“消息回溯”能力，则消费者已消费，且确认删除的消息，会立即删除，开启该功能时，须指定回溯的“可回溯周期”。	“可回溯周期”的范围，必须小于等于消息的生命周期。建议将回溯周期与消息的生命周期设置为相同的值，便于定位问题。
可回溯时间范围	若开启“消息回溯”能力，则消费者确认删除的消息不会立即删除，会持续保存到此处配置的最大时间。	时间范围：1天 - 15天，设置较长可能会产生昂贵的存储费用。最大可回溯时间点 = 当前时间 - 设置的可回溯时间范围。消息生产时间在这个值之前的不可回溯。
可回溯存储空间	开启回溯消息后，如果持久存储的消息超过此最大存储空间，则会从后向前删除（优先删除旧数据）。	存储空间范围：1GB - 10GB，设置较大可能会产生昂贵的存储费用。

3. 单击**提交**，在队列服务列表可以看到创建好的队列服务。

步骤2：使用 SDK 收发消息

> **说明**
> 以下示例以 Java 语言客户端说明，其他语言客户端接入请参见 SDK 文档 。

1. 下载 Demo 并解压。
2. 引入 CMQ 客户端相关依赖

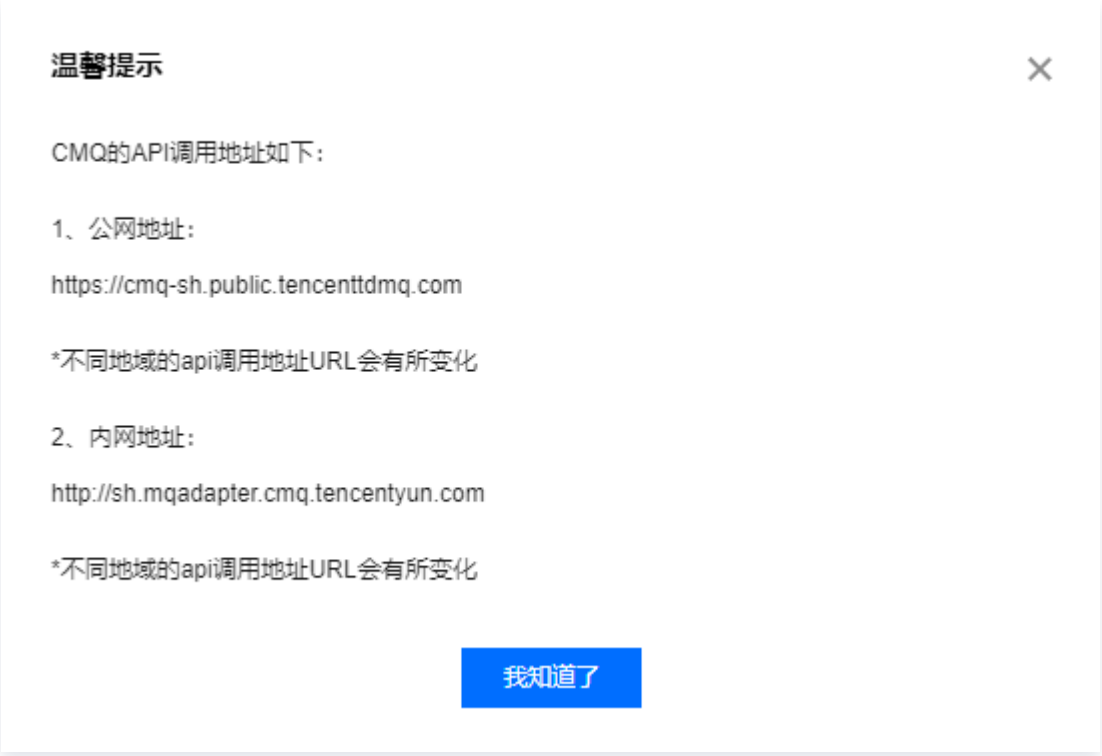
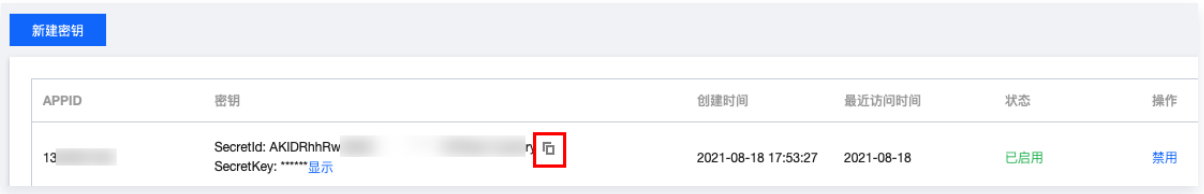
```
<!-- cmq sdk -->
<dependency>
    <groupId>com.qcloud</groupId>
    <artifactId>cmq-http-client</artifactId>
    <version>1.0.7</version>
</dependency>

<!-- 云API sdk -->
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java</artifactId>
    <version>3.1.423</version>
```

```
</dependency>
```

3. 发送消息

```
Account account = new Account(SERVER_ENDPOINT, SECRET_ID, SECRET_KEY);
Queue queue = account.getQueue(queueName);
String msg = "hello client, this is a message. Time:" + new Date();
CmqResponse response = queue.send(msg);
```

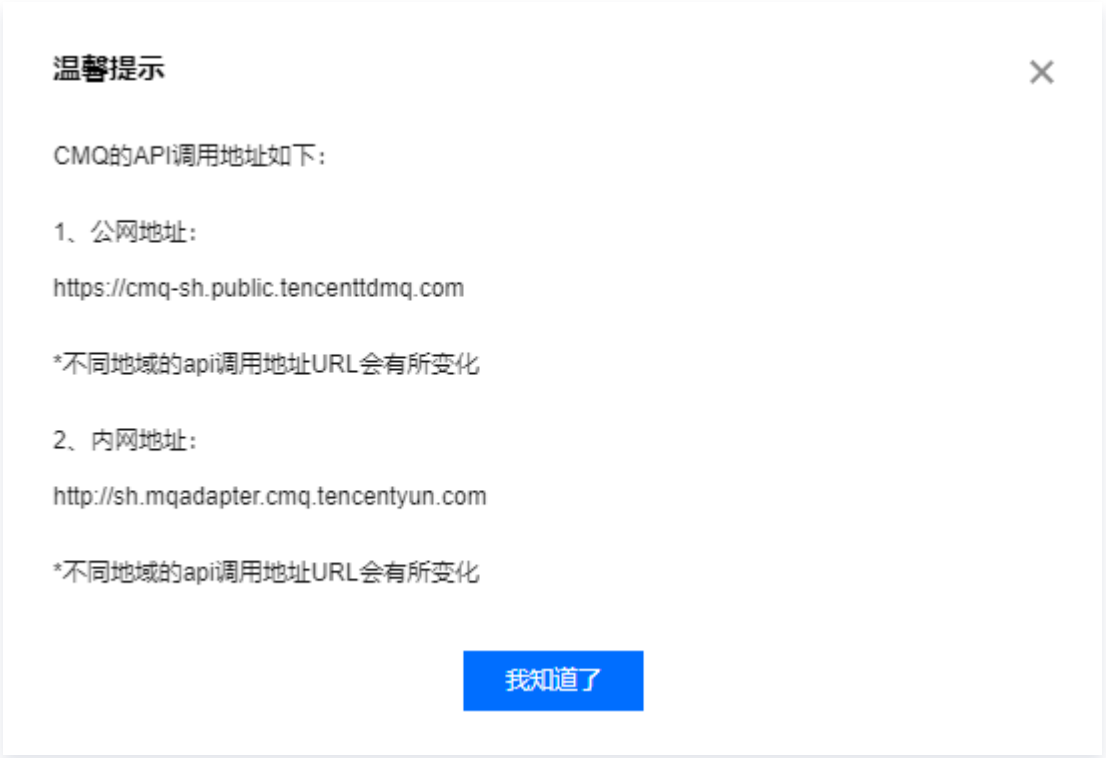
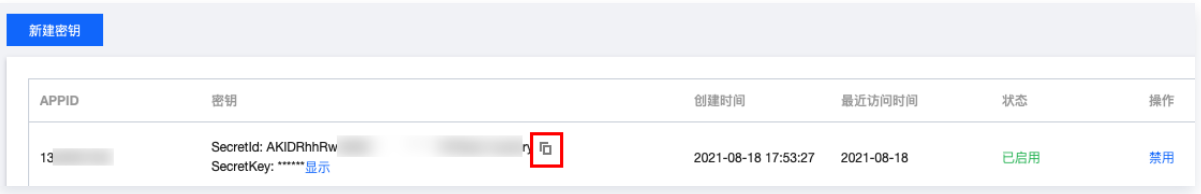
参数	说明
SERVER_ENDPOINT	API 调用地址，在 TDMQ CMQ 版控制台 的队列服务 > API 请求地址处复制。 
SECRET_ID、SE	云 API 密钥，登录 访问管理控制台，在访问密钥 > API 密钥管理页面复制。 

CRET_KEY	
queueName	队列名称，在 TDMQ CMQ 版控制台 的队列服务列表页面获取。

4. 消费消息。

```
    Account account = new Account(SERVER_ENDPOINT, SECRET_ID,
SECRET_KEY);
    Queue queue = account.getQueue(queueName);
    Message message = queue.receiveMessage();
    // 消费成功，删除消息。未删除的消息，将在一定时间后可重新投递
    queue.deleteMessage(message.receiptHandle);
```

参数	说明

SERVER_ENDPOINT	API 调用地址，在 TDMQ CMQ 版控制台 的队列服务 > API 请求地址处复制。 
SECRET_ID、SECRET_KEY	云 API 密钥，登录 访问管理控制台 ，在访问密钥 > API 密钥管理页面复制。 
queu	队列名称，在 TDMQ CMQ 版控制台 的队列服务列表页面获取。

e N a m e

说明

以上是 CMQ 的生产和消费方式的简单介绍，更多操作可参见 Demo 。

主题模型快速入门

最近更新时间：2024-10-14 10:59:51

操作场景

本文为您介绍从零开始创建一个主题并使用 Java SDK 进行收发消息测试的方法，帮助您快速了解客户端接入 TDMQ CMQ 版所需的基本操作。

前提条件

已 注册腾讯云账号 。

操作步骤

步骤1：创建主题

1. 登录 TDMQ CMQ 版 控制台。
2. 在左侧导航栏选择**主题订阅**，选择好地域，单击**新建**，填写主题名称。

- 主题名称：以字母起始，只能包含字母、数字、“-”及“_”，最大64字符，创建后不能修改，不区分大小写。
- 消息堆积：未触发推送到订阅者，或订阅者接收失败的消息，暂时堆积到主题中。

- 消息过滤类型：
  - 标签：CMQ 提供生产、订阅的消息标签匹配能力，可用于消息过滤。详细规则参见 标签键匹配功能说明 。
  - 路由匹配：Binding key、Routing key 是组合使用的，完全兼容 rabbitmq topic 匹配模式。发消息时配的 Routing key 是客户端发消息带的。创建订阅关系时配的 Binding key 是 topic 和 订阅者 的绑定关系。详细规则请参见 路由键匹配功能说明 。
- 资源标签：选填，标签可以帮助您从各种维度方便地对 TDMQ CMQ 版资源进行分类管理，具体使用方法可参见 标签管理 。

3. 单击**提交**，在主题订阅列表可以看到创建好的主题。

步骤2：创建订阅

主题发布消息有一个前提，即需要有订阅者订阅主题，如果没有订阅者存在，那么主题中的消息不会被投递，此时发布消息这一操作就失去了意义。

1. 在 主题订阅 页面，单击刚刚创建的主题的“ID”，进入主题详情页面。
2. 选择页面上方的**订阅者**页签，单击**新建**，填写订阅者相关信息。

- 订阅者类型
  - Queue 队列服务：订阅者可以填写一个 Queue，使用队列来接收发布的消息。
  - URL地址：订阅者也可以不与 Queue 结合，自己来处理消息。详情请参见 投递消息 。
- 添加订阅者标签：添加订阅者时，需增加 FilterTag。增加 FilterTag 后，该订阅者仅能收到带该 FilterTag 的消息，单个订阅者最多可添加5个 tag。只要其中某个 tag 能匹配 Topic 的过滤标签，订阅者即可收到该次 Topic 投递的消息，若消息不带任何标签，则该订阅者无法收到该类型消息。
  - 标签：详细规则参见 标签键匹配功能说明 。
  - 路由匹配：详细规则请参见 路由键匹配功能说明 。
- 重试策略：主题发布消息之后，会自动将消息推送给订阅，当推送失败时，有两种重试策略：
  - **退避重试**：重试3次，间隔时间为10s – 20s之间的一个随机值，超过3次后，该条消息对于该订阅者丢弃，不会再重试。
  - **衰退指数重试**：重试176次，总计重试时间为1天，间隔时间依次为：2^0，2^1，…，512，512，…，512秒。默认为衰退指数重试策略。

3. 单击**提交**，在订阅者列表可以看到刚刚创建好的订阅者。

步骤3：使用 SDK 收发消息

> **说明**
> 以下示例以 Java 语言客户端说明，其他语言客户端接入请参见 SDK 文档 。

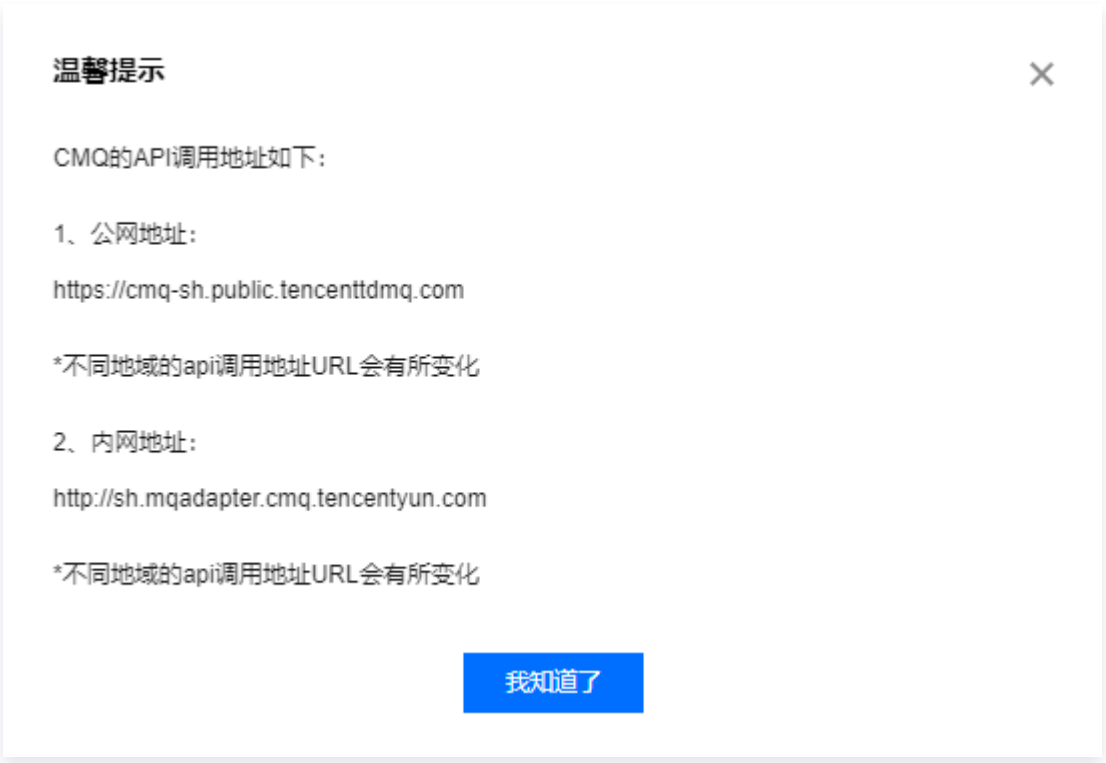
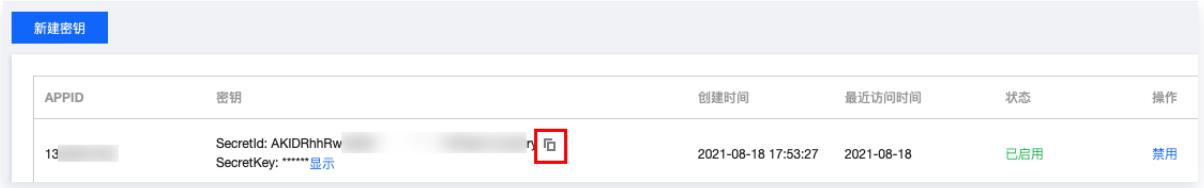
1. 下载 Demo 并解压。
2. 引入cmq客户端相关依赖。

```
 <!-- cmq sdk -->
<dependency>
    <groupId>com.qcloud</groupId>
    <artifactId>cmq-http-client</artifactId>
    <version>1.0.7</version>
</dependency>

<!-- 云API sdk -->
<dependency>
    <groupId>com.tencentcloudapi</groupId>
    <artifactId>tencentcloud-sdk-java</artifactId>
    <version>3.1.423</version>
</dependency>
```

3. 创建Topic对象。

```
Account account = new Account(SERVER_ENDPOINT, SECRET_ID,
SECRET_KEY);
Topic topic = account.getTopic(topicName);
```

参数	说明
SERVER_ENDPOINT	API 调用地址，在 TDMQ CMQ 版控制台 的主题订阅 > API 请求地址处复制。 
SECRET_ID、SECRET_KEY	云 API 密钥，登录 访问管理控制台 ，在访问密钥 > API 密钥管理页面复制。 
topicName	主题订阅名称，在 TDMQ CMQ 版控制台 的主题订阅列表页面获取。

4. 发送 TAG 类型消息。

```
        String msg = "hello client, this is a message. tag=TAG1. Time:"
+ new Date();
        List<String> tags = Collections.singletonList("TAG1");
        String messageId = topic.publishMessage(msg, tags, null);
```

5. 发送 route 消息。

```
        String msg = "hello client, this is a message. route(abc) Time:"
+ new Date();
        String messageId = topic.publishMessage(msg, "abc");
```

6. 消费消息，使用订阅者对应的queue进行消费。

```
        Account account = new Account(SERVER_ENDPOINT, SECRET_ID,
SECRET_KEY);
        Queue queue = account.getQueue(queueName);
        Message message = queue.receiveMessage();
        // 消费成功，删除消息。未删除的消息，将在一定时间后可重新投递
        queue.deleteMessage(message.receiptHandle);
```

> **说明**
> 以上是 CMQ 的生产和消费方式的简单介绍，更多操作可参见 Demo 。