消息队列 CMQ 版

SDK 文档

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

SDK 文档
HTTP 数据流 SDK
API 概述

最近更新时间：2024-10-14 16:39:52

TDMQ CMQ 版目前支持 Java、Python、PHP 及 C++ SDK，后续会支持更多语言。欢迎广大开发者根据 API 说明开发更多语言版本的 SDK。
为保证消息队列 CMQ 用户迁移至 TDMQ CMQ 版无需修改业务代码，以下接口协议与以前版本保持一致。

接口功能	Action ID	功能描述
发送消息	SendMessage	用于发送一条消息到指定的队列。
批量发送消息	BatchSendMessage	用于发送批量消息到指定的队列。
消费消息	ReceiveMessage	用于消费队列中的一条消息。
批量消费消息	BatchReceiveMessage	用于消费队列中的多条消息。
删除消息	DeleteMessage	用于删除已经被消费过的消息。
批量删除消息	BatchDeleteMessage	用于批量删除已经被消费过的消息。
发布消息	PublishMessage	用于发布一条消息到指定主题。
批量发布消息	BatchPublishMessage	用于发布批量消息到指定主题。

除此之外的其他接口，需要按照 HTTP 控制流 SDK 的引导进行开发。

Java SDK

最近更新时间：2024-12-17 17:42:22

操作场景

本文以 Java SDK 为例介绍客户端接入 TDMQ CMQ 版服务并收发消息的操作步骤。

前提条件

- 安装1.8或以上版本 JDK
- 安装2.5或以上版本 Maven
- 下载 Demo

队列模型

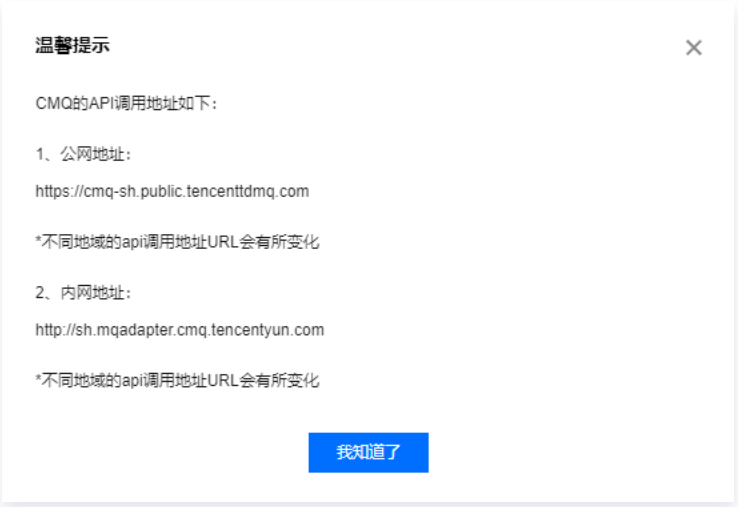
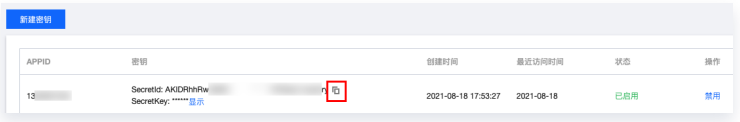
操作步骤

1. 在控制台创建符合需求的队列服务，参见 创建队列服务 。
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>3.1.498</version>
</dependency>
```

3. 发送消息。

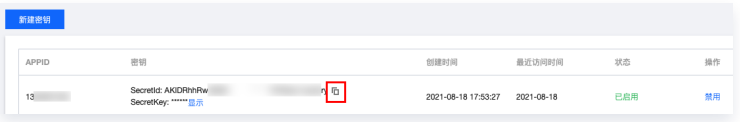
```
   Account account = new Account(SERVER_ENDPOINT, SECRET_ID,
SECRET_KEY);
   Queue queue = account.getQueue(queueName);
   String msg = "hello client, this is a message. Time:" + new Date();
   CmqResponse response = queue.send(msg);
```

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

主题模型

操作步骤

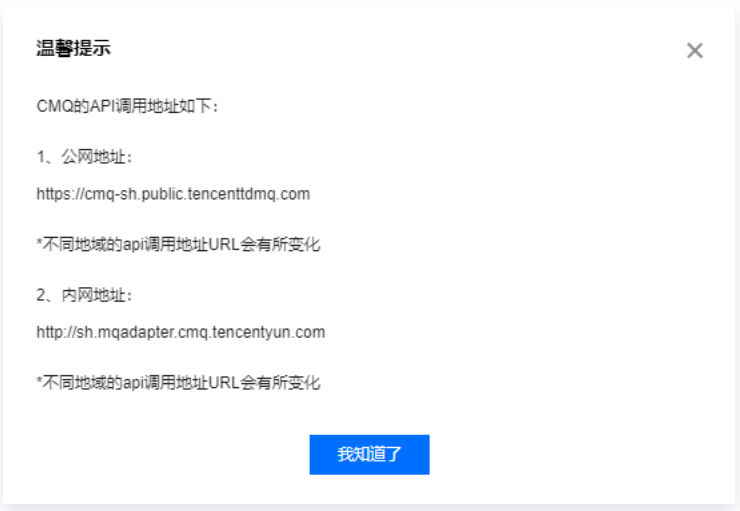
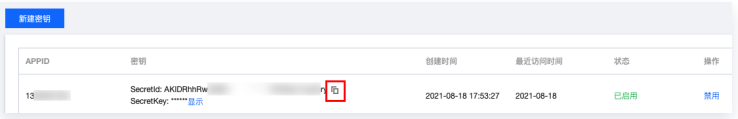
1. 在控制台创建资源。
   1.1 在控制台创建主题，参见 主题管理 。
   1.2 给主题创建一个订阅者，参见 订阅管理 。
2. 引入 CMQ 客户端相关依赖。

```
<!-- 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。 
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);
```

4. 发送 route 消息。

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

5. 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 两种模型下的生产和消费方式的简单介绍，更多使用可参见 Demo 。

Python SDK

最近更新时间：2024-12-17 17:42:22

操作步骤

本文以 Python SDK 为例介绍客户端接入 TDMQ CMQ 版服务并收发消息的操作步骤。

前提条件

- 安装 Python
- 安装 pip
- 下载 Demo

队列模型

操作步骤

1. 在控制台创建符合需求的队列，参见 创建队列服务 。

> **说明**
> 创建消息队列可在控制台手动创建，或通过云 API 进行创建，使用云 API 需要安装相关 SDK，SDK 安装可参见 Python SDK 安装 。

shell

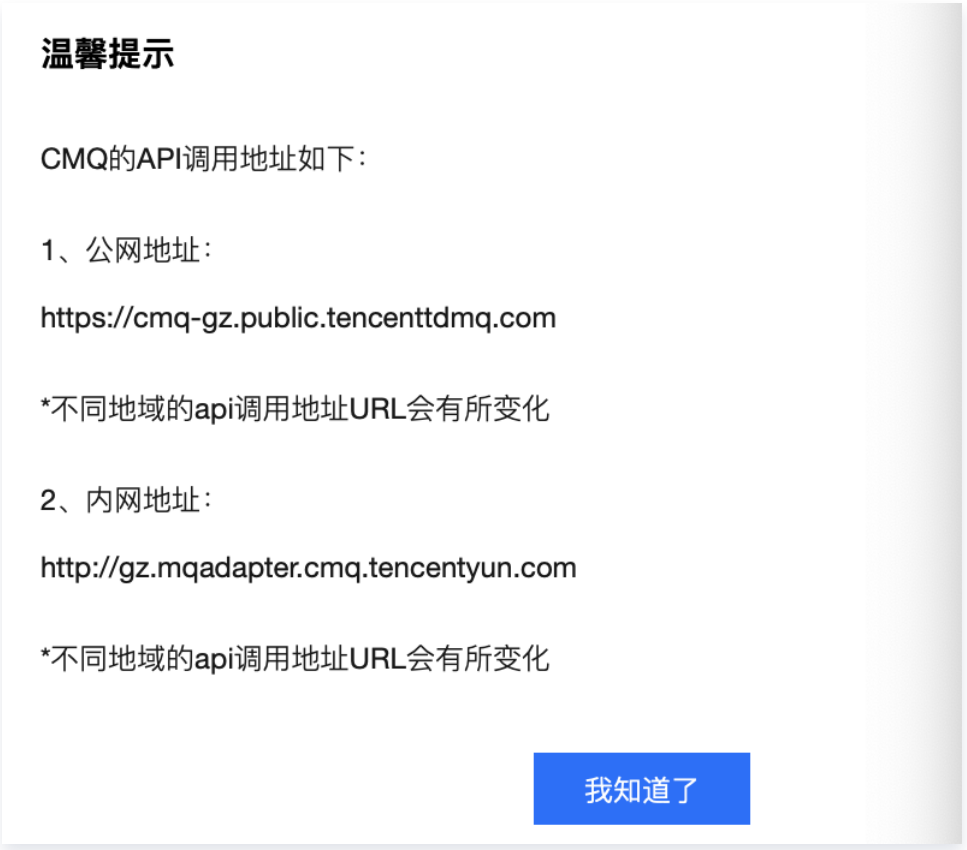
```
pip install --upgrade tencentcloud-sdk-python
```

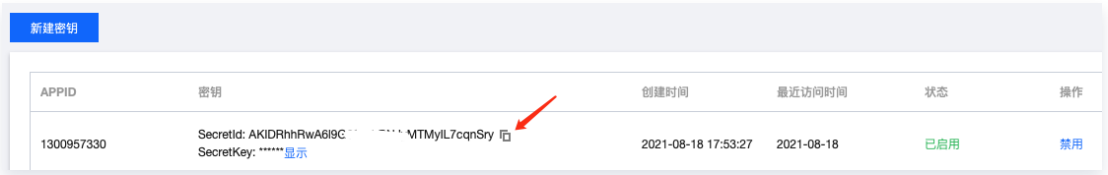
python

```
# api认证信息
cred = credential.Credential(SecretId, SecretKey)
httpProfile = HttpProfile()
httpProfile.endpoint = NameServerAddress

clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
# 创建tdmq客户端
client = tdmq_client.TdmqClient(cred, "ap-guangzhou", clientProfile)

# 创建cmq队列请求参数
```

```
req = models.CreateCmqQueueRequest()
params = {
    "QueueName": "queue_api",
    # 下面是死信队列相关配置
    "DeadLetterQueueName": "dead_queue_api",  # 死信队列，该消息队列要先创建
    "Policy": 0,  # 0为消息被多次消费未删除，1为Time-To-Live过期
    "MaxReceiveCount": 3  # 最大接收次数 1-1000
}
req.from_json_string(json.dumps(params))

# 创建cmq消息队列
resp = client.CreateCmqQueue(req)
```

参数	说明
NameServerAddress	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。

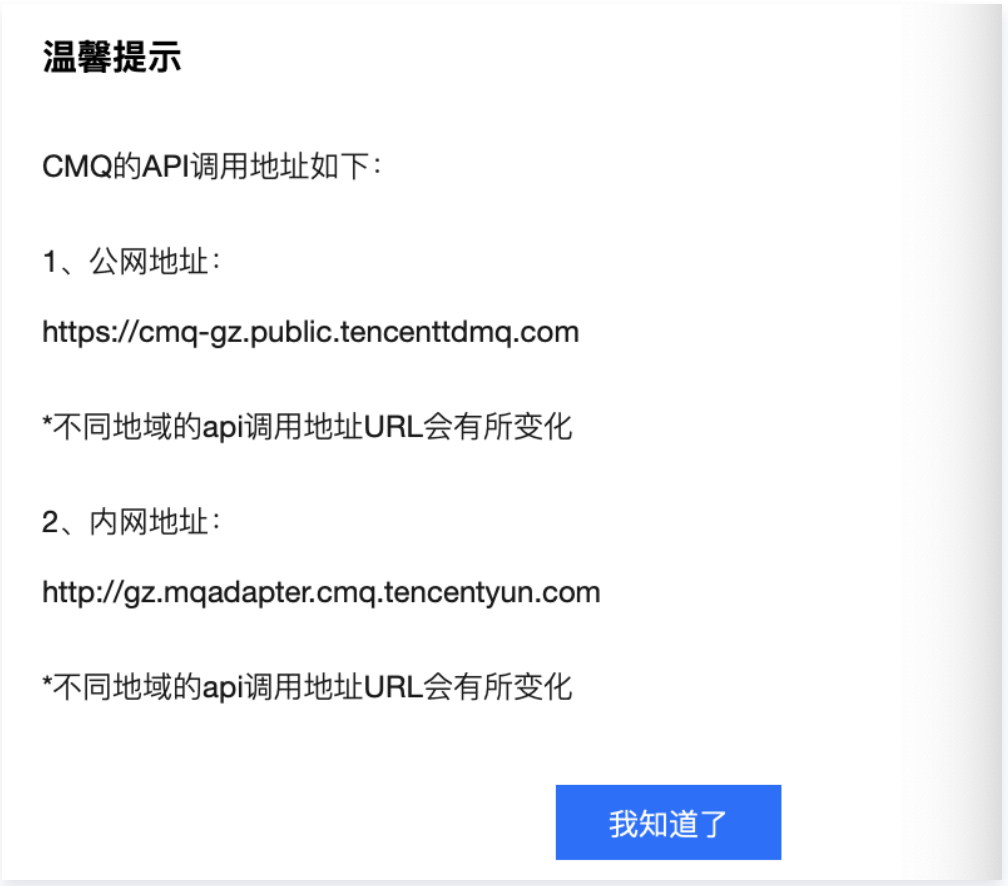
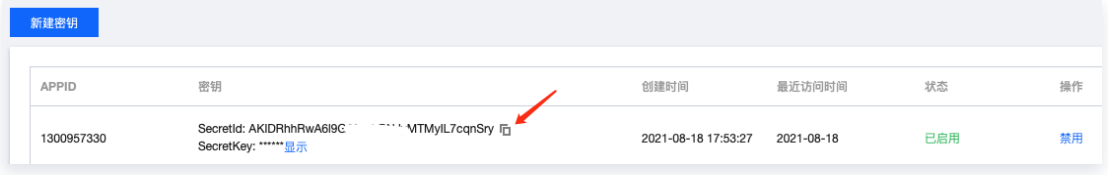


SecretId、SecretKey	云 API 密钥，登录 访问管理控制台，在访问密钥 > API密钥管理页面复制。 

2. 在项目中引入 CMQ 相关文件，需要根据使用的 Python 版本选择分支，默认为 Python2 SDK，您可切换至 Python3 分支中查看 Python3 SDK。
3. 发送消息。

```
import os
import sys

sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)) + "/..")

import logging
from cmq.account import Account
from cmq.queue import Message
from cmq.cmq_exception import CMQExceptionBase

# 腾讯云账户 secretId、secretKey，此处还需注意密钥对的保密
# 密钥可前往https://console.cloud.tencent.com/cam/capi网站进行获取
secretId = 'AKIDSiiRtxxxx'
secretKey = 'GGzSeaM5xxxx'
# CMQ的服务调用地址
nameServerAddress = 'https://cmq-gz.public.tencenttdmq.com'

# 初始化 my_account, my_queue
# Account类对象不是线程安全的，如果多线程使用，需要每个线程单独初始化Account类对象
my_account = Account(nameServerAddress, secretId, secretKey,
debug=True)
my_account.set_log_level(logging.DEBUG)
# 消息队列名称
queue_name = sys.argv[1] if len(sys.argv) > 1 else "python_queue"
my_queue = my_account.get_queue(queue_name)

try:
    # 消息内容
    msg_body = "I am test message."
    msg = Message(msg_body)
    # 发送消息
```

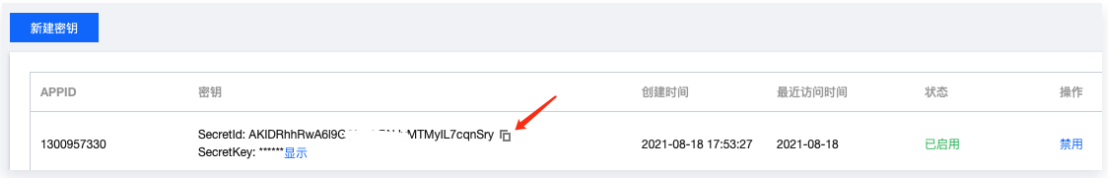
```
        re_msg = my_queue.send_message(msg)
        # 发送结果
        print("Send Message Succeed! MessageBody:%s MessageID:%s" %
(msg_body, re_msg.msgId))
except CMQExceptionBase as e:
        print("Send Message Fail! Exception:%s\n" % e)
```

参数	说明
NameServerAddress	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。 
SecretId、SecretKey	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。 
queue_name	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。

```
import os
import sys

sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)) + "/..")

import logging
from cmq.account import Account
from cmq.cmq_exception import CMQExceptionBase

# 腾讯云账户 secretId、secretKey, 此处还需注意密钥对的保密
# 密钥可前往https://console.cloud.tencent.com/cam/capi网站进行获取
secretId = 'AKIDSiiRtxxxx'
secretKey = 'GGzSeaM5xxxx'
# CMQ的服务调用地址
nameServerAddress = 'https://cmq-gz.public.tencenttdmq.com'

# 初始化 my_account, my_queue
# Account类对象不是线程安全的，如果多线程使用，需要每个线程单独初始化Account类对
象
my_account = Account(nameServerAddress, secretId, secretKey,
debug=True)
my_account.set_log_level(logging.DEBUG)
queue_name = sys.argv[1] if len(sys.argv) > 1 else "python_queue"
my_queue = my_account.get_queue(queue_name)

try:
    wait_seconds = 3
    # 获取消息
    recv_msg = my_queue.receive_message(wait_seconds)
    # 具体业务
    print("Receive Message Succeed! ReceiptHandle:%s MessageBody:%s
MessageID:%s" % (
            recv_msg.receiptHandle, recv_msg.msgBody,
recv_msg.msgId))
    # 消费成功，删除消息
    my_queue.delete_message(recv_msg.receiptHandle)
except CMQExceptionBase as e:
    print("Receive Message Fail! Exception:%s\n" % e)
```

参数	说明

NameServerAddress	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。 
SecretId、SecretKey	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。 
queue	队列名称，在 TDMQ CMQ 版控制台 的队列服务列表页面获取。

主题模型

操作步骤

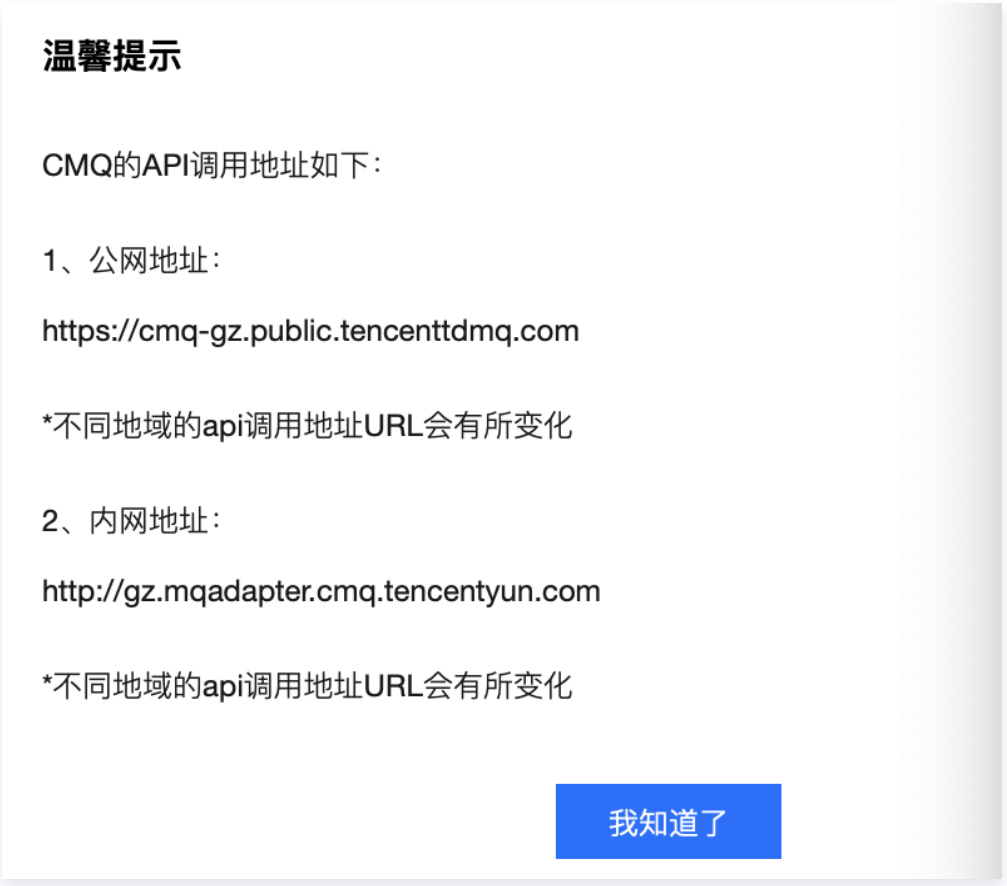
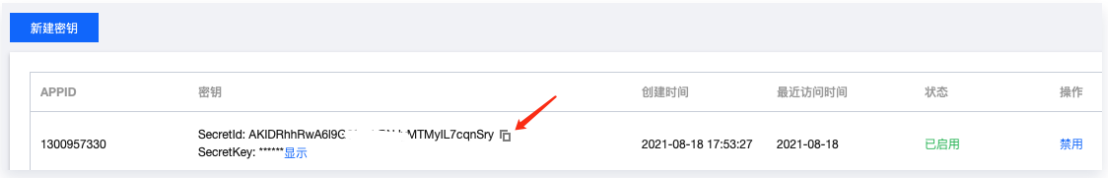
1. 准备所需资源，创建主题订阅和订阅者。

   1.1 创建主题订阅。可通过控制台手动创建，也可以通过云 API 进行创建，使用云 API 需要安装相关 SDK，SDK 安装可参见 Python SDK 安装 。

```
# api认证信息
cred = credential.Credential(SecretId, SecretKey)
```

```
httpProfile = HttpProfile()
httpProfile.endpoint = NameServerAddress

clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
client = tdmq_client.TdmqClient(cred, "ap-guangzhou",
clientProfile)

req = models.CreateCmqTopicRequest()
params = {
        "TopicName": "topic_api",  # 主题名字，在单个地域同一帐号下唯一
        "FilterType": 1,  # 用于指定主题的消息匹配策略。1：表示标签匹配策
略；2：表示路由匹配策略
        "MsgRetentionSeconds": 86400  # 消息保存时间。取值范围60 -
86400 s（即1分钟 - 1天）
}
req.from_json_string(json.dumps(params))

# 创建topic
resp = client.CreateCmqTopic(req)
```

参数	说明

NameServerAddress	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。 
SecretId、SecretKey	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。 

1.2 创建订阅者。可通过控制台进行手动创建，也可以通过云 API 进行创建，使用云 API 需要安装相关 SDK，SDK 安装可参见 Python SDK 安装 。

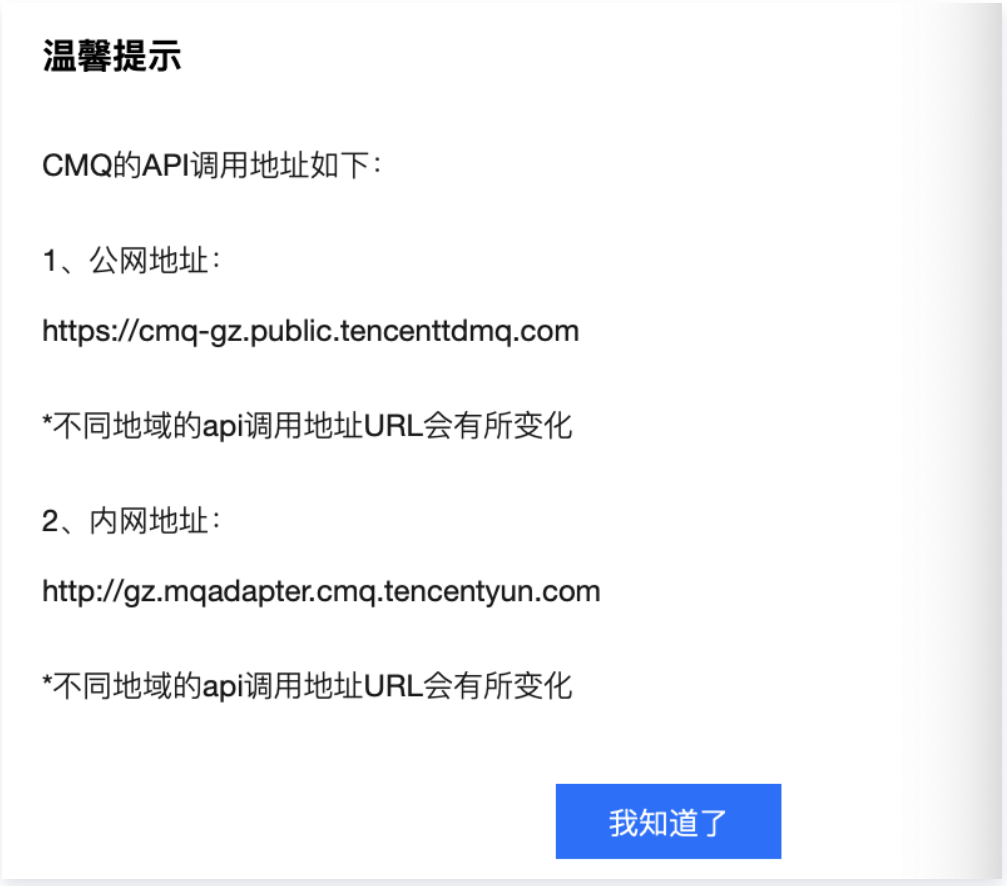
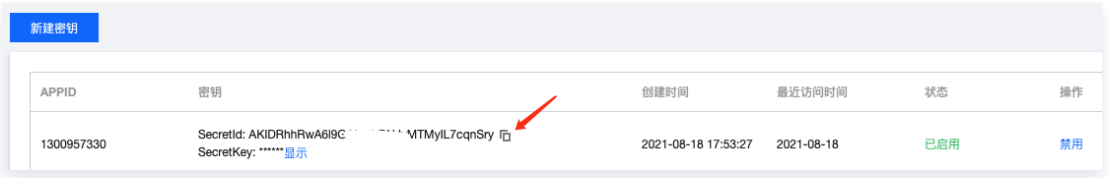
```
# api认证信息
cred = credential.Credential(SecretId, SecretKey)
httpProfile = HttpProfile()
httpProfile.endpoint = NameServerAddress

clientProfile = ClientProfile()
clientProfile.httpProfile = httpProfile
client = tdmq_client.TdmqClient(cred, "ap-guangzhou",
clientProfile)
```

```
req = models.CreateCmqSubscribeRequest()
params = {
        "TopicName": "topic_api",  # 创建订阅的topic名称
        "SubscriptionName": "sub",  # 订阅名称
        "Protocol": "queue",  # 订阅的协议，目前支持两种协议：http、queue。使用http协议，用户需自己搭建接受消息的web server。使用queue，消息会自动推送到CMQ queue，用户可以并发地拉取消息。
        "Endpoint": "topic_queue_api",   # 接收通知的Endpoint，根据协议Protocol区分：对于http，Endpoint必须以“http://”开头，host可以是域名或IP；对于Queue，则填QueueName。
        "NotifyStrategy": "BACKOFF_RETRY",  # CMQ推送服务器的重试策略。取值有：1）BACKOFF_RETRY，退避重试。；2）EXPONENTIAL_DECAY_RETRY，指数衰退重试。
        "FilterTag": ["TAG"],  # 消息标签（用于消息过滤）。标签数量不能超过5个
        # "BindingKey": ["a.b.c"],  # BindingKey数量不超过5个， 每个BindingKey长度不超过64字节，该字段表示订阅接收消息的过滤策略
        "NotifyContentFormat": "SIMPLIFIED"  # 推送内容的格式。取值：1）JSON；2）SIMPLIFIED，即raw格式。如果Protocol是queue，则取值必须为SIMPLIFIED。如果Protocol是http，两个值均可以，默认值是JSON。
}
req.from_json_string(json.dumps(params))

# 创建订阅
resp = client.CreateCmqSubscribe(req)
```

⚠ 注意

BindingKey 与 FilterTag 要根据所订阅topic类型进行设置，否则无效。

参数	说明

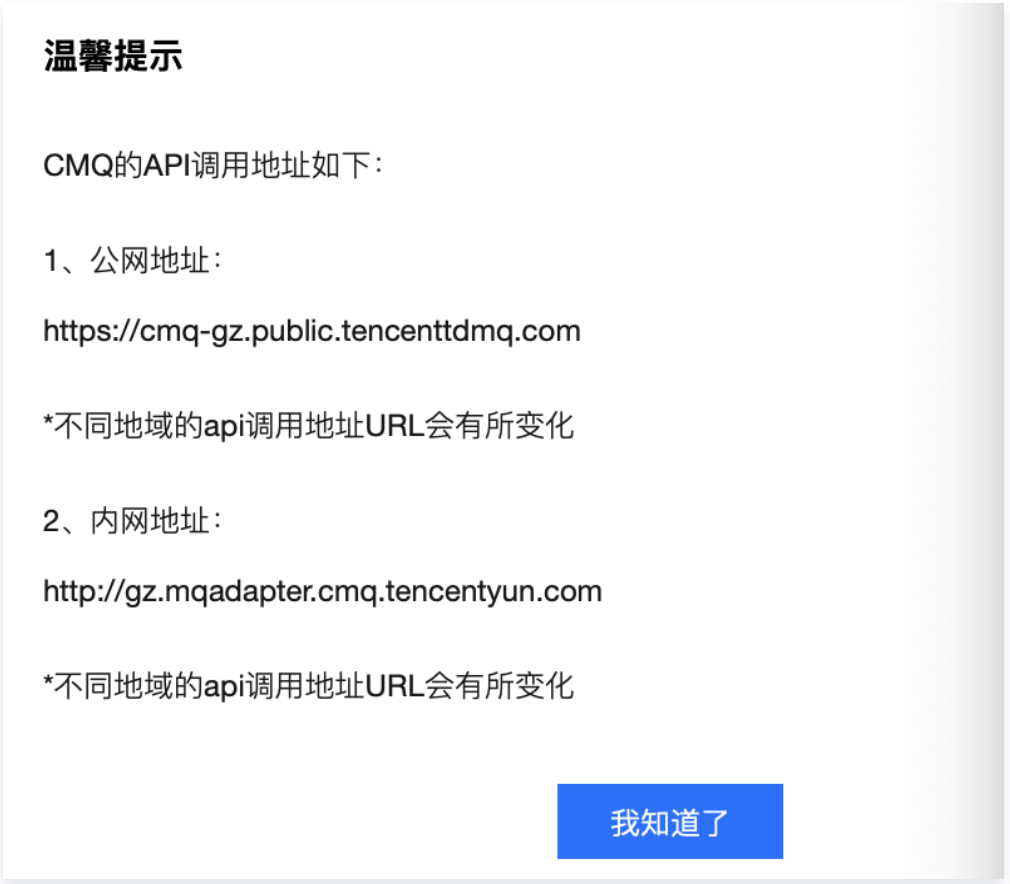
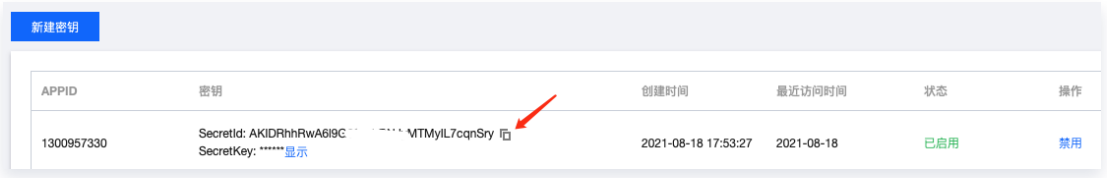
NameServerAddress	API 调用地址，在 TDMQ CMQ 版控制台 的队列服务 > API 请求地址处复制。 
SecretId、SecretKey	云 API 密钥，登录 访问管理控制台，在访问密钥 > API 密钥管理页面复制。 

2. 在项目中引入 CMQ 相关文件，需要根据使用的 Python 版本选择分支，默认为 Python2 SDK，您可切换至 Python3 分支中查看 Python3 SDK。
3. 创建 my_topic，用来发布消息。

```
import os
import sys

sys.path.insert(0, os.path.dirname(os.path.abspath(__file__)) + "/..")

import logging
from cmq.account import Account
from cmq.cmq_exception import *
from cmq.topic import *
```

```
# 腾讯云账户 secretId、secretKey，此处还需注意密钥对的保密
# 密钥可前往https://console.cloud.tencent.com/cam/capi网站进行获取
secretId = 'AKIDSiiRtxxxx'
secretKey = 'GGzSeaM5xxxx'
# CMQ的服务调用地址
nameServerAddress = 'https://cmq-gz.public.tencenttdmq.com'

try:
    # 初始化 my_account
    # Account类对象不是线程安全的，如果多线程使用，需要每个线程单独初始化
Account类对象
    my_account = Account(nameServerAddress, secretId, secretKey,
debug=True)
    my_account.set_log_level(logging.DEBUG)
    # topic主题名称
    topic_name = sys.argv[1] if len(sys.argv) > 1 else
"python_topic_route"
    my_topic = my_account.get_topic(topic_name)
except CMQExceptionBase as e:
    print("Exception:%s\n" % e)
```

参数	说明

NameServerAddress	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。 
SecretId、SecretKey	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。 
topic_name	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。

```
# 消息tag
tags = ["TAG", "TAG1", "TAG2"]
for tag in tags:
        # 发送tag消息
```

```
        message = Message("this is a test TAG message. TAG:" + tag,
[tag])
        re_msg = my_topic.publish_message(message)
        # 发送结果
        print("Send Message Succeed! MessageBody:%s MessageID:%s" %
(message.msgBody, re_msg.msgId))
```

5. 发送 route 消息。

```
# 消息route信息
routes = ["a.b.c", "a.b.x", "a.c.d", "x.y.z", "x.y.c"]
for route in routes:
    message = Message("this is a test route message. Route:" + route)
# 发送route消息
re_msg = my_topic.publish_message(message, route)
# 发送结果
print("Send Message Succeed! MessageBody:%s MessageID:%s" %
(message.msgBody, re_msg.msgId))
```

6. 消费者消费订阅者订阅的消息队列即可。

> **说明**
> 以上是 CMQ 两种模型下的生产和消费方式的简单介绍，更多使用可参见 Demo 或 CMQ 代码仓库 。

HTTP 控制流 SDK

最近更新时间：2024-08-06 11:14:11

操作场景

本文主要介绍消息队列 CMQ 版控制流接口列表和 SDK 使用方式。数据流（收发消息相关）接口请参见 HTTP 数据流 API 概述 。

> **注意：**
> 云 API SendCmqMsg 和 PublishCmqMsg 仅用于进行控制台发送队列和主题模式下的测试消息。因为云 API 存在限流，因此在真实业务场景下，请使用数据流 SDK 进行消息收发。

消息队列 CMQ 版控制流接口列表如下：

接口名称	接口功能
CreateCmqQueue	创建 TDMQ CMQ 版队列接口
CreateCmqSubscribe	创建 TDMQ CMQ 版订阅接口
CreateCmqTopic	创建 TDMQ CMQ 版主题
DeleteCmqQueue	删除 TDMQ CMQ 版队列
DeleteCmqSubscribe	删除 TDMQ CMQ 版订阅
DeleteCmqTopic	删除 TDMQ CMQ 版主题
DescribeCmqDeadLetterSourceQueues	枚举 TDMQ CMQ 版死信队列源队列
DescribeCmqQueueDetail	查询 TDMQ CMQ 版队列详情
DescribeCmqQueues	查询 TDMQ CMQ 版全量队列
DescribeCmqSubscriptionDetail	查询 TDMQ CMQ 版订阅详情
DescribeCmqTopicDetail	查询 TDMQ CMQ 版主题详情
DescribeCmqTopics	枚举 TDMQ CMQ 版全量主题
ModifyCmqQueueAttribute	修改 TDMQ CMQ 版队列属性
ModifyCmqSubscriptionAttribute	修改 TDMQ CMQ 版订阅属性
ModifyCmqTopicAttribute	修改 TDMQ CMQ 版主题属性

RewindCmqQueue	回溯 TDMQ CMQ 版队列
UnbindCmqDeadLetter	解绑 TDMQ CMQ 版死信队列

操作步骤

消息队列 CMQ 版控制流 SDK 使用新版 API，以**创建 TDMQ CMQ 版队列接口**为例，具体使用方式如下。

1. 登录 云 API 控制台 。
2. 选择 **API Explore > 分布式消息队列 > CMQ 管理相关接口**。
3. 选择具体的接口，并填写输入参数。输入参数说明如下：

参数名称	描述
QueueName	队列名字，在单个地域同一账号下唯一。队列名称是一个不超过 64 个字符的字符串，必须以字母为首字符，剩余部分可以包含字母、数字和横划线(-)。
MaxMsgHeapNum	最大堆积消息数。取值范围在公测期间为 1,000,000 - 10,000,000，正式上线后范围可达到 1000,000-1000,000,000。默认取值在公测期间为 10,000,000，正式上线后为 100,000,000。
PollingWaitSeconds	消息接收长轮询等待时间。取值范围 0-30 秒，默认值 0。
VisibilityTimeout	消息可见性超时。取值范围 1-43200 秒（即12小时内），默认值 30。
MaxMsgSize	消息最大长度。取值范围 1024-65536 Byte（即1-64K），默认值 65536。
MsgRetentionSeconds	消息保留周期。取值范围 60-1296000 秒（1min-15天），默认值 345600 (4天)。
RewindSeconds	队列是否开启回溯消息能力，该参数取值范围0-msgRetentionSeconds，即最大的回溯时间为消息在队列中的保留周期，0表示不开启。
Transaction	1 表示事务队列，0 表示普通队列。
FirstQueryInterval	第一次回查间隔。
MaxQueryCount	最大回查次数。
DeadLetterQueueName	死信队列名称。
Policy	死信策略。0为消息被多次消费未删除，1为 Time-To-Live 过期。

MaxReceiveCount	最大接收次数 1-1000。
MaxTimeToLive	polic 为1时必选。最大未消费过期时间。范围300-43200，单位秒，需要小于消息最大保留时间msgRetentionSeconds。
Trace	是否开启消息轨迹追踪，当不设置字段时，默认为不开启，该字段为 true 表示开启，为 false 表示不开启。

4. 在线调用。输入参数填写完成后，在页面上方选择**在线调用**页签，单击**发送请求**，返回结果如下。

5. 生成代码示例。验证完成后，在**代码生成**页签的代码框中选择自己需要的语言，即可生成对应的代码示例。以 Java 语言为例：

```
import com.tencentcloudapi.common.Credential;
import com.tencentcloudapi.common.profile.ClientProfile;
import com.tencentcloudapi.common.profile.HttpProfile;
import com.tencentcloudapi.common.exception.TencentCloudSDKException;
import com.tencentcloudapi.tdmq.v20200217.TdmqClient;
import com.tencentcloudapi.tdmq.v20200217.models.*;

public class CreateCmqQueue
{
    public static void main(String [] args) {
```

```
        try{
            // 实例化一个认证对象，入参需要传入腾讯云账户 secretId，
secretKey,此处还需注意密钥对的保密
            // 密钥可前往https://console.cloud.tencent.com/cam/capi网站进
行获取
            Credential cred = new Credential("SecretId", "SecretKey");
            // 实例化一个 http 选项，可选的，没有特殊需求可以跳过
            HttpProfile httpProfile = new HttpProfile();
            httpProfile.setEndpoint("tdmq.tencentcloudapi.com");
            // 实例化一个client选项，可选的，没有特殊需求可以跳过
            ClientProfile clientProfile = new ClientProfile();
            clientProfile.setHttpProfile(httpProfile);
            // 实例化要请求产品的 client 对象,clientProfile 是可选的
            TdmqClient client = new TdmqClient(cred, "ap-guangzhou",
clientProfile);
            // 实例化一个请求对象,每个接口都会对应一个 request 对象
            CreateCmqQueueRequest req = new CreateCmqQueueRequest();
            req.setQueueName("queen");
            req.setPollingWaitSeconds(10L);
            req.setVisibilityTimeout(10L);
            req.setMaxMsgSize(1048576L);
            req.setMsgRetentionSeconds(345600L);
            // 返回的 resp 是一个 CreateCmqQueueResponse 的实例，与请求对象
对应
            CreateCmqQueueResponse resp = client.CreateCmqQueue(req);
            // 输出 json 格式的字符串回包

System.out.println(CreateCmqQueueResponse.toJsonString(resp));
        } catch (TencentCloudSDKException e) {
            System.out.println(e.toString());
        }
    }
}
```

SDK 参数配置说明

最近更新时间：2025-05-16 14:14:22

消息队列 CMQ 版（TDMQ CMQ 版）是一种分布式消息队列服务，它能够提供可靠的，基于消息的异步通信机制，能够将分布式部署的不同应用（或同一应用的不同组件）中的信息传递，存储在可靠有效的 TDMQ CMQ 版队列中，防止消息丢失。TDMQ CMQ 版支持多进程同时读写，收发互不干扰，无需各应用或组件始终处于运行状态。
TDMQ CMQ 版提供了四种 SDK，本文以 Python 为例进行说明。

Python SDK 简介

为了方便使用，TDMQ CMQ 版将用户操作、队列操作、主题操作等抽象成了几个类：

- Account：封装账户 SecretId、SecretKey，用户可以创建删除队列、主题及订阅，并查看这些对象。
- queue：收发消息，查看设置队列属性。
- topic：发布消息，查看设置主题属性，查看订阅者。
- cmq_client：可以设置一些客户端的与服务器端连接属性，如设置是否写日志、连接超时时间、是否长连接等。

> **注意**
> 所有的类均为非线程安全的，如果需要多线程使用，最好每个线程实例化自己的对象。

下载 SDK >>

队列模型

这里的队列与数据结构中定义的 Queue 有一定区别。数据结构中的队列严格按照 FIFO 操作，这里的分布式队列不会有严格的 FIFO（后续会推出专属的 FIFO 产品）。这里的队列相当于一个高性能、高容量、高可靠的容器，可以生产消息投递进去，也可以把消息取出来消费。队列在初始化的时候有自己的属性设置。属性以及含义如下：

属性	描述
maxMsgHeapNum	最大堆积消息数。队列中可以存储的消息条数，该属性表示了队列的存储和堆积能力。
pollingWaitSeconds	消息接收长轮询等待时间。取值范围0 – 30秒，该时间设置表示的是消费消息时默认等待接收消息的时间。 例如设置为10，那么在消费消息时，如果没有消息， 默认会等待10s返回；如果有消息则会立即返回。 也可以在接收消息的时候设置自定义的等待时间，不使用队列的属性值。
visibilityTimeout	消息可见性超时。

	消息被消费者获取到，会有一个不可见时间，即在这个时间之内别的消费者无法获取这条消息。取值范围1 - 43200秒（即12小时内），默认值为30。
maxMsgSize	消息最大长度。取值范围1024 - 1048576 Byte（即1K - 1024K），默认值为65536。
msgRetentionSeconds	消息生命周期，即消息在队列中的保存时间，取值范围 60 - 43200秒（1分钟 - 12小时），SDK 创建时默认值为 3600（1小时）。
createTime	队列的创建时间。返回 Unix 时间戳，精确到秒。
lastModifyTime	最后一次修改队列属性的时间。返回 Unix 时间戳，精确到秒。
activeMsgNum	在队列中处于 Active 状态（不处于被消费状态）的消息总数，为近似值。
inactiveMsgNum	在队列中处于 Inactive 状态（正处于被消费状态）的消息总数，为近似值。
rewindSeconds	回溯队列的消息回溯时间最大值，取值范围0 - 43200秒，0表示不开启消息回溯。
rewindmsgNum	已调用 DelMsg 接口删除但还在回溯保留时间内的消息数量。
minMsgTime	消息最小未消费时间，单位为秒。
delayMsgNum	延时消息数量。

查看队列模型快速入门>>

主题模型

主题模型类似设计模式中的发布订阅模式，Topic 相当于发布消息的单位，Topic 下面的订阅者相当于观察者模式。Topic 会把发布的消息主动推送给订阅者：

属性	描述
msgCount	当前该主题中堆积的消息数目（消息堆积数）。
maxMsgSize	消息最大长度。取值范围1024 - 1048576Byte（即1 - 1024KB），默认值为65536。
msgRetentionSeconds	消息在主题中最长存活时间，从发送到该主题开始经过此参数指定的时间后，不论消息是否被成功推送给用户都将被删除，该参数单位为秒。固定为一天（86400秒），该属性不能修改。

createTime	主题的创建时间。返回 Unix 时间戳，精确到秒。
lastModifyTime	最后一次修改主题属性的时间。返回 Unix 时间戳，精确到秒。
filterType	描述用户创建订阅时选择的过滤策略： • filterType = 1 表示用户使用 filterTag 标签过滤。 • filterType = 2 表示用户使用 bindingKey 过滤。