分布式消息服务 RocketMQ 版

API 参考

文档版本 06

发布日期 2023-12-15

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
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。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 使用前必读

1.1 概述

欢迎使用分布式消息服务RocketMQ版。分布式消息服务RocketMQ版是一个低延迟、弹性高可靠、高吞吐、动态扩展、便捷多样的消息中间件服务。

本文档提供了分布式消息服务RocketMQ版API的描述、语法、参数说明及样例等内容。

须知

分布式消息服务RocketMQ版持续增加新的功能，将不可避免对现有接口有所调整，比如增加响应参数。

为了减少接口变更带来的影响，除了分布式消息服务RocketMQ版自身尽量做到接口向下兼容的同时，用户在使用过程中，应当接受出现返回内容（JSON格式）含有未使用的参数和值的现象，即能够正常忽略未使用的参数和值。

1.2 调用说明

分布式消息服务RocketMQ版提供了REST（Representational State Transfer）风格API，支持您通过HTTPS请求调用，调用方法请参见**如何调用API**。

1.3 终端节点

终端节点（Endpoint）即调用API的**请求地址**，不同服务不同区域的终端节点不同，您可以从**地区和终端节点**中查询所有服务的终端节点。

1.4 约束与限制

- 您能创建的实例数上限，具体请参见**服务配额**。
- 更详细的限制请参见具体API的说明。

1.5 基本概念

- 账号

  用户注册账号时，账号对其所拥有的资源及云服务具有完全的访问权限，可以重置用户密码、分配用户权限等。由于账号是付费主体，为了确保账号安全，建议您不要直接使用账号进行日常管理工作，而是创建用户并使用他们进行日常管理工作。

- 用户

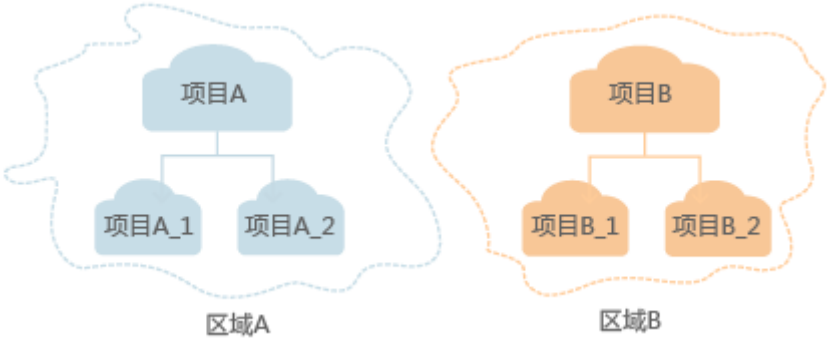
  由账号在IAM中创建的用户，是云服务的使用人员，具有身份凭证（密码和访问密钥）。

  通常在调用API的鉴权过程中，您需要用到账号、用户和密码等信息。

- 区域：指云资源所在的物理位置，同一区域内可用区间内网互通，不同区域间内网不互通。通过在不同地区创建云资源，可以将应用程序设计的更接近特定客户的要求，或满足不同地区的法律或其他要求。
- 可用区：一个可用区是一个或多个物理数据中心的集合，有独立的风火水电，AZ内逻辑上再将计算、网络、存储等资源划分成多个集群。一个Region中的多个AZ间通过高速光纤相连，以满足用户跨AZ构建高可用性系统的需求。
- 项目

  区域默认对应一个项目，这个项目由系统预置，用来隔离物理区域间的资源（计算资源、存储资源和网络资源），以默认项目为单位进行授权，用户可以访问您账号中该区域的所有资源。如果您希望进行更加精细的权限控制，可以在区域默认的项目中创建子项目，并在子项目中创建资源，然后以子项目为单位进行授权，使得用户仅能访问特定子项目中资源，使得资源的权限控制更加精确。

  图 1-1 项目隔离模型

- 企业项目

  企业项目是项目的升级版，针对企业不同项目间资源的分组和管理，是逻辑隔离。企业项目中可以包含多个区域的资源，且项目中的资源可以迁入迁出。

  关于企业项目ID的获取及企业项目特性的详细信息，请参见《**企业管理服务用户指南**》。

2 API 概览

表 2-1 实例管理类接口

API	说明
生命周期管理	包括创建实例、修改实例信息、查询实例、删除实例、查询代理列表。
消费组管理	包括查询消费组、创建消费组、删除消费组、修改消费组、查询消费列表或详情、重置消费进度。
Topic管理	包括创建主题、删除主题、查询主题、修改主题、查询主题消费组列表、查询主题的消息数。
消息管理	包括查询消息、查询消息轨迹、导出死信消息、重发死信消息、消费验证。
用户管理	包括创建用户、查询用户、修改用户、删除用户、查询主题的授权用户列表、查询消费组的授权用户列表。
元数据迁移	包括新建元数据迁移任务、查询实例下所有迁移任务或查询指定迁移任务信息、删除元数据迁移任务。
参数管理	包括查询RocketMQ配置和修改RocketMQ配置。
标签管理	包括添加或删除实例标签、查询实例标签和查询项目标签。
其他接口	包括查询可用区信息。
规格变更管理	包括查询实例的扩容规格列表和实例规格变更。

3 如何调用 API

3.1 构造请求

本节介绍REST API请求的组成，并以调用IAM服务的**获取用户Token**来说明如何调用API，该API获取用户的Token，Token可以用于调用其他API时鉴权。

请求 URI

请求URI由如下部分组成。

{URI-scheme}://{Endpoint}/{resource-path}?{query-string}

尽管请求URI包含在请求消息头中，但大多数语言或框架都要求您从请求消息中单独传递它，所以在此单独强调。

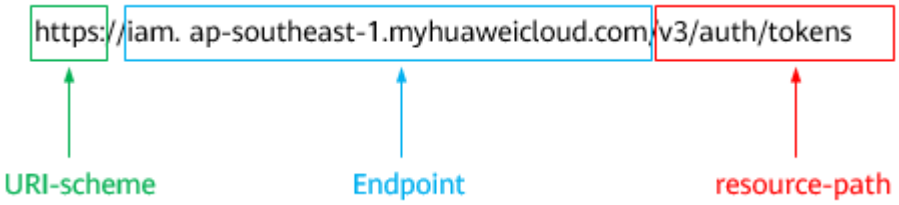
表 3-1 URI 中的参数说明

参数	描述
URI-scheme	表示用于传输请求的协议，当前所有API均采用**HTTPS**协议。
Endpoint	指定承载REST服务端点的服务器域名或IP，不同服务不同区域的Endpoint不同，您可以从**地区和终端节点**获取。 例如IAM服务在“中国-香港”区域的Endpoint为“iam.ap-southeast-1.myhuaweicloud.com”。
resource-path	资源路径，即API访问路径。从具体API的URI模块获取，例如“获取用户Token”API的resource-path为“/v3/auth/tokens”。
query-string	查询参数，是可选部分，并不是每个API都有查询参数。查询参数前面需要带一个“?”，形式为“参数名=参数取值”，例如“?limit=10”，表示查询不超过10条数据。

例如您需要获取IAM在“中国-香港”区域的Token，则需使用“中国-香港”区域的Endpoint（iam.ap-southeast-1.myhuaweicloud.com），并在**获取用户Token**的URI部分找到resource-path（/v3/auth/tokens），拼接起来如下所示。

```
https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
```

图 3-1 URI 示意图



说明

为查看方便，在每个具体API的URI部分，只给出resource-path部分，并将请求方法写在一起。这是因为URI-scheme都是HTTPS，而Endpoint在同一个区域也相同，所以简洁起见将这两部分省略。

请求方法

HTTP请求方法（也称为操作或动词），它告诉服务你正在请求什么类型的操作。

- **GET**：请求服务器返回指定资源。
- **PUT**：请求服务器更新指定资源。
- **POST**：请求服务器新增资源或执行特殊操作。
- **DELETE**：请求服务器删除指定资源，如删除对象等。
- **HEAD**：请求服务器资源头部。
- **PATCH**：请求服务器更新资源的部分内容。当资源不存在的时候，PATCH可能会去创建一个新的资源。

在**获取用户Token**的URI部分，您可以看到其请求方法为“POST”，则其请求为：

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
```

请求消息头

附加请求头字段，如指定的URI和HTTP方法所要求的字段。例如定义消息体类型的请求头“Content-Type”，请求鉴权信息等。

详细的公共请求消息头字段请参见**表3-2**。

表 3-2 公共请求消息头

名称	描述	是否必选	示例
Host	请求的服务器信息，从服务API的URL中获取。值为hostname[:port]。端口缺省时使用默认的端口，https的默认端口为443。	否 使用AK/SK认证时该字段必选。	code.test.com or code.test.com:443

名称	描述	是否必选	示例
Content-Type	消息体的类型（格式）。推荐用户使用默认值application/json，有其他取值时会在具体接口中说明。	是	application/json
Content-Length	请求body长度，单位为Byte。	否	3495
X-Project-Id	project id，项目编号。请参考**获取项目ID**章节获取项目编号。	否 如果是专属云场景采用AK/SK认证方式的接口请求，或者多project场景采用AK/SK认证的接口请求，则该字段必选。	e9993fc787d94b6c886cbaa340f9c0f4
X-Auth-Token	用户Token。 用户Token也就是调用**获取用户Token**接口的响应值，该接口是唯一不需要认证的接口。 请求响应成功后在响应消息头（Headers）中包含的“X-Subject-Token”的值即为Token值。	否 使用Token认证时该字段必选。	注：以下仅为Token示例片段。 MIIPAgYJKoZIhvcNAQcCo...ggg1BBIINPXsidG9rZ

说明

API同时支持使用AK/SK认证，AK/SK认证是使用SDK对请求进行签名，签名过程会自动往请求中添加Authorization（签名认证信息）和X-Sdk-Date（请求发送的时间）请求头。

AK/SK认证的详细说明请参见**认证鉴权**的“AK/SK认证”。

对于**获取用户Token**接口，由于不需要认证，所以只添加“Content-Type”即可，添加消息头后的请求如下所示。

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json
```

请求消息体（可选）

该部分可选。请求消息体通常以结构化格式（如JSON或XML）发出，与请求消息头中Content-type对应，传递除请求消息头之外的内容。若请求消息体中参数支持中文，则中文字符必须为UTF-8编码。

每个接口的请求消息体内容不同，也并不是每个接口都需要有请求消息体（或者说消息体为空），GET、DELETE操作类型的接口就不需要消息体，消息体具体内容需要根据具体接口而定。

对于**获取用户Token**接口，您可以从接口的请求部分看到所需的请求参数及参数说明。将消息体加入后的请求如下所示，加粗的斜体字段需要根据实际值填写，其中***username***为用户名，***domainname***为用户所属的账号名称，*********为用户登录密码，***xxxxxxxxxxxxxxxxx***为project的名称，如“ap-southeast-1”，您可以从**地区和终端节点**获取。

说明

scope参数定义了Token的作用域，上面示例中获取的Token仅能访问project下的资源。您还可以设置Token作用域为某个账号下所有资源或账号的某个project下的资源，详细定义请参见**获取用户Token**。

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/tokens
Content-Type: application/json

{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxxxxxxxxxxx"
            }
        }
    }
}
```

到这里为止这个请求需要的内容就具备齐全了，您可以使用**curl**、**Postman**或直接编写代码等方式发送请求调用API。对于获取用户Token接口，返回的响应消息头中“x-subject-token”就是需要获取的用户Token。有了Token之后，您就可以使用Token认证调用其他API。

3.2 认证鉴权

调用接口有如下两种认证方式，您可以选择其中一种进行认证鉴权。

- Token认证：通过Token认证调用请求。
- AK/SK认证：通过AK（Access Key ID）/SK（Secret Access Key）加密调用请求。推荐使用AK/SK认证，其安全性比Token认证要高。

Token 认证

> **说明**
>
> Token的有效期为24小时，需要使用一个Token鉴权时，可以先缓存起来，避免频繁调用。

Token在计算机系统中代表令牌（临时）的意思，拥有Token就代表拥有某种权限。Token认证就是在调用API的时候将Token加到请求消息头，从而通过身份认证，获得操作API的权限。Token可通过调用**获取用户Token**接口获取。

云服务存在两种部署方式：项目级服务和全局级服务。其中：

- 项目级服务需要获取项目级别的Token，此时请求body中**auth.scope**的取值为**project**。
- 全局级服务需要获取全局级别的Token，此时请求body中**auth.scope**的取值为**domain**。

调用本服务API需要project级别的Token，即调用**获取用户Token**接口时，请求body中**auth.scope**的取值需要选择**project**，如下所示。

```
{
    "auth": {
        "identity": {
            "methods": [
                "password"
            ],
            "password": {
                "user": {
                    "name": "username",
                    "password": "********",
                    "domain": {
                        "name": "domainname"
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxx"
            }
        }
    }
}
```

获取Token后，再调用其他接口时，您需要在请求消息头中添加“X-Auth-Token”，其值即为Token。例如Token值为“ABCDEFJ....”，则调用接口时将“X-Auth-Token: ABCDEFJ....”加到请求消息头即可，如下所示。

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3/auth/projects
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

AK/SK 认证

> **说明**
>
> AK/SK签名认证方式仅支持消息体大小12M以内，12M以上的请求请使用Token认证。

AK/SK认证就是使用AK/SK对请求进行签名，在请求时将签名信息添加到消息头，从而通过身份认证。

- AK(Access Key ID)：访问密钥ID。与私有访问密钥关联的唯一标识符；访问密钥ID和私有访问密钥一起使用，对请求进行加密签名。

- SK(Secret Access Key)：与访问密钥ID结合使用的密钥，对请求进行加密签名，可标识发送方，并防止请求被修改。

使用AK/SK认证时，您可以基于签名算法使用AK/SK对请求进行签名，也可以使用专门的签名SDK对请求进行签名。详细的签名方法和SDK使用方法请参见**API签名指南**。

须知

签名SDK只提供签名功能，与服务提供的SDK不同，使用时请注意。

3.3 返回结果

状态码

请求发送以后，您会收到响应，包含状态码、响应消息头和消息体。

状态码是一组从1xx到5xx的数字代码，状态码表示了请求响应的状态，完整的状态码列表请参见**状态码**。

对于**获取用户Token**接口，如果调用后返回状态码为“201”，则表示请求成功。

响应消息头

对应请求消息头，响应同样也有消息头，如“Content-type”。

对于**获取用户Token**接口，返回如**图3-2**所示的消息头，其中“x-subject-token”就是需要获取的用户Token。有了Token之后，您就可以使用Token认证调用其他API。

图 3-2 获取用户 Token 响应消息头

响应消息体（可选）

该部分可选。响应消息体通常以结构化格式（如JSON或XML）返回，与响应消息头中Content-type对应，传递除响应消息头之外的内容。

对于**获取用户Token**接口，返回如下消息体。为篇幅起见，这里只展示部分内容。

```
{
    "token": {
        "expires_at": "2019-02-13T06:52:13.855000Z",
        "methods": [
            "password"
        ],
        "catalog": [
            {
                "endpoints": [
                    {
                        "region_id": "XXXXXX",
.....
```

当接口调用出错时，会返回错误码及错误信息说明，错误响应的Body体格式如下所示。

```
{
    "error_msg": "The format of message is error",
    "error_code": "AS.0001"
}
```

其中，error_code表示错误码，error_msg表示错误描述信息。

4 快速入门

场景描述

您可以根据业务需要创建相应计算能力和存储空间的RocketMQ实例。

API调用方法请参考**如何调用API**。

前提条件

已获取IAM和RocketMQ的Endpoint，具体请参见**地区和终端节点**。

创建 RocketMQ 实例

如下示例是创建RocketMQ实例的请求消息：

```
{
 "name": "rocketmq-test",
 "engine": "reliability",
 "engine_version": "4.8.0",
 "storage_space": 600,
 "vpc_id": "ead6cxxxxxx9ba91820e72c",
 "subnet_id": "3cb6axxxxxxa7f671d6a8",
 "security_group_id": "d39c8xxxxxxaa8510a498",
 "available_zones": [
  "effdcxxxxxx2f56533"
 ],
 "product_id": "c6.4u8g.cluster",
 "storage_spec_code": "dms.physical.storage.high.v2",
 "broker_num": 1
}
```

- name：实例名称，由您自行定义。
- engine：消息引擎，设置为reliability。
- engine_version：消息引擎的版本。
- storage_space：消息存储空间，单位MB。具体取值范围，请参考**创建实例**。
- vpc_id：RocketMQ实例所在的VPC（虚拟私有云）的ID。您可以在**虚拟私有云控制台**查询，也可以通过**查询VPC列表**API查询。
- subnet_id：VPC内子网的网络ID。您可以在**虚拟私有云控制台**查询，也可以通过**查询子网列表**API查询。
- security_group_id：安全组ID。您可以在**虚拟私有云控制台**查询，也可以通过**查询安全组列表**API查询。

- available_zones：创建节点到指定的AZ ID，该参数不能为空数组或者数组的值为空，请参考**查询可用区信息**获取。
- product_id：产品规格。具体取值范围，请参考**创建实例**。
- storage_spec_code：存储IO规格。具体取值范围，请参考**创建实例**。
- broker_num：代理个数。

5 API V2（推荐）

5.1 生命周期管理

5.1.1 查询所有实例列表

功能介绍

查询租户的实例列表，支持按照条件查询。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances

表 5-1 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。

表 5-2 Query 参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎：rocketmq。 缺省值：**rocketmq**
name	否	String	实例名称。

参数	是否必选	参数类型	描述
instance_id	否	String	实例ID。
status	否	String	实例状态，详细状态说明请参考**实例状态说明**。
include_failure	否	String	是否返回创建失败的实例数。 当参数值为“true”时，返回创建失败的实例数。参数值为“false”或者其他值，不返回创建失败的实例数。
exact_match_name	否	String	是否按照实例名称进行精确匹配查询。 默认为“false”，表示模糊匹配实例名称查询。若参数值为“true”表示按照实例名称进行精确匹配查询。
enterprise_project_id	否	String	企业项目ID。
limit	否	Integer	当次查询返回的最大个数，默认值为10，取值范围为1~50。
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。

请求参数

无

响应参数

状态码： 200

表 5-3 响应 Body 参数

参数	参数类型	描述
instances	Array of **ShowInstanceResp** objects	实例列表。
instance_num	Integer	实例数量。

表 5-4 ShowInstanceResp

参数	参数类型	描述
name	String	实例名称。
engine	String	引擎。
status	String	状态。
description	String	消息描述。
type	String	实例类型：集群，cluster。
specification	String	实例规格。
engine_version	String	版本。
instance_id	String	实例ID。
charging_mode	Integer	付费模式，1表示按需计费，0表示包年/包月计费。
vpc_id	String	私有云ID。
vpc_name	String	私有云名称。
created_at	String	完成创建时间。 格式为时间戳，指从格林威治时间1970年01月01日00时00分00秒起至指定时间的偏差总毫秒数。
product_id	String	产品标识。
security_group_id	String	安全组ID。
security_group_name	String	租户安全组名称。
subnet_id	String	子网ID。
subnet_name	String	子网名称。
subnet_cidr	String	子网路由（仅RocketMQ 5.x版本会显示此字段）。
available_zones	Array of strings	可用区ID列表。
available_zone_names	Array of strings	可用区名称列表。
user_id	String	用户ID。
user_name	String	用户名。
maintain_begin	String	维护时间窗开始时间，格式为HH:mm:ss。

参数	参数类型	描述
maintain_end	String	维护时间窗结束时间，格式为HH:mm:ss。
enable_log_collection	Boolean	是否开启消息收集功能。
storage_space	Integer	存储空间，单位：GB。
used_storage_space	Integer	已用消息存储空间，单位：GB。
enable_publicip	Boolean	是否开启公网。
publicip_id	String	实例绑定的弹性IP地址的ID。以英文逗号隔开多个弹性IP地址的ID。如果开启了公网访问功能（即enable_publicip为true），该字段为必选。
publicip_address	String	公网IP地址。
ssl_enable	Boolean	是否开启SSL。
cross_vpc_info	String	跨VPC访问信息。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	存储规格代码。
service_type	String	服务类型。
storage_type	String	存储类型。
extend_times	Long	扩展时间。
ipv6_enable	Boolean	是否开启IPv6。
support_features	String	实例支持的特性功能。
disk_encrypted	Boolean	是否开启磁盘加密。
ces_version	String	云监控版本。
node_num	Integer	节点数。
new_spec_billing_enable	Boolean	是否启用新规格计费。
enable_acl	Boolean	是否开启访问控制列表。
broker_num	Integer	节点数（仅RocketMQ 4.8.0版本会显示此字段）。

参数	参数类型	描述
namesrv_address	String	元数据地址。
broker_address	String	业务数据地址。
public_namesrv_address	String	公网元数据地址。
public_broker_address	String	公网业务数据地址。
grpc_address	String	grpc连接地址（仅RocketMQ 5.x版本会显示此字段）。
public_grpc_address	String	公网grpc连接地址（仅RocketMQ 5.x版本会显示此字段）。
enterprise_project_id	String	企业项目ID。
tags	Array of **TagEntity** objects	标签列表。
total_storage_space	Integer	总存储空间。
resource_spec_code	String	资源规格。

表 5-5 TagEntity

参数	参数类型	描述
key	String	标签键。 ● 不能为空。 ● 对于同一个实例，Key值唯一。 ● 长度为1~128个字符（中文也可以输入128个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ ● 首尾字符不能为空格。

参数	参数类型	描述
value	String	标签值。 ● 长度为0~255个字符（中文也可以输入255个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ ● 首尾字符不能为空格。

请求示例

查询所有实例的列表。

```
GET https://{endpoint}/v2/{project_id}/instances?engine=reliability
```

响应示例

状态码： 200

查询所有实例列表成功。

- ```
  {
    "instances" : [ {
      "name" : "reliability-ztest",
      "engine" : "reliability",
      "status" : "RUNNING",
      "description" : "",
      "type" : "cluster",
      "specification" : "c6.4u8g.cluster * 1 broker",
      "engine_version" : "4.8.0",
      "instance_id" : "68fdc9a8-805e-439d-8dd9-25adc1c58bf6",
      "resource_spec_code" : "",
      "charging_mode" : 1,
      "vpc_id" : "3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf",
      "vpc_name" : "vpc-1101840",
      "created_at" : "1636699753874",
      "product_id" : "c6.4u8g.cluster",
      "security_group_id" : "23c5977f-ff33-4b95-a73e-08d8a0bc4e6c",
      "security_group_name" : "Sys-default",
      "subnet_id" : "0a0f1fcb-f019-458d-b9e5-301867394d50",
      "available_zones" : [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
      "available_zone_names" : [ "AZ3" ],
      "user_id" : "0b01fbb53600d4671fa8c00673c71260",
      "user_name" : "",
      "maintain_begin" : "02:00:00",
      "maintain_end" : "06:00:00",
      "enable_log_collection" : false,
      "storage_space" : 558,
      "total_storage_space" : 600,
      "used_storage_space" : 28,
      "enable_publicip" : false,
      "ssl_enable" : false,
      "cross_vpc_info" : "{\"192.168.1.21\":{\"advertised_ip\":\"192.168.1.21\",\"broker_port\":\"-
  \",\"port_id\":\"fa020857-d899-497c-a5f8-8dd90ed67ff7\",\"namesrv_port
  \":\"8301\"},\"192.168.1.246\":{\"advertised_ip\":\"192.168.1.246\",\"broker_port\":\"10100\",\"port_id
  \":\"c0f0586f-a4ee-41b5-a7f1-b4e554bbf12d\",\"namesrv_port\":\"-\"},\"192.168.1.77\":
  {\"advertised_ip\":\"192.168.1.77\",\"broker_port\":\"10101\",\"port_id
  \":\"8eb1d144-4315-402f-8498-37c9a10b630c\",\"namesrv_port\":\"-\"},\"192.168.1.159\":
  {\"advertised_ip\":\"192.168.1.159\",\"broker_port\":\"-\",\"port_id\":\"d35b22af-a202-4329-
  bda4-26e1bdf2aa8e\",\"namesrv_port\":\"8300\"},\"192.168.1.14\":{\"advertised_ip
  \":\"192.168.1.14\",\"broker_port\":\"10102\",\"port_id\":\"5fcb50fd-3af5-4123-a455-
  ```
  ```

```
a67f1b926026\",\"namesrv_port\":\"-\"}}",
    "storage_resource_id" : "164bdaef-2e67-4fd5-be8b-a18f91d455a2",
    "storage_spec_code" : "dms.physical.storage.ultra.v2",
    "service_type" : "advanced",
    "storage_type" : "hec",
    "enterprise_project_id" : "0",
    "extend_times" : 0,
    "ipv6_enable" : false,
    "support_features" :
"kafka.crossvpc.domain.enable,feature.physerver.kafka.topic.accesspolicy,rabbitmq.plugin.management,
rocketmq.acl,roma_app_enable,auto_topic_switch,feature.physerver.kafka.user.manager,enable.new.aut
hinfo,route,kafka.config.dynamic.modify.enable,kafka.new.pod.port,feature.physerver.kafka.topic.modify
,message_trace_enable,features.pod.token.access,kafka.ssl.cert.modify.enable,roma.user.manage.no.sup
port,feature.physerver.kafka.pulbic.dynamic,features.log.collection,kafka.config.static.modify.enable",
    "disk_encrypted" : false,
    "ces_version" : "linux,v1,v2",
    "node_num" : 5,
    "new_spec_billing_enable" : true,
    "broker_num" : 1,
    "namesrv_address" : "***",
    "broker_address" : "***",
    "public_namesrv_address" : "***",
    "public_broker_address" : "***",
    "grpc_address" : "***",
    "public_grpc_address" : "***"
  } ],
  "instance_num" : 1
}
```


SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListInstancesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running
this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
```

```
        ListInstancesRequest request = new ListInstancesRequest();
        try {
            ListInstancesResponse response = client.listInstances(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListInstancesRequest()
        response = client.list_instances(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
```

```
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListInstancesRequest{}
    response, err := client.ListInstances(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询所有实例列表成功。

错误码

请参见**错误码**。

5.1.2 创建实例

功能介绍

创建实例，该接口支持创建按需和包周期两种计费方式的实例。

调用方法

请参见**如何调用API**。

URI

POST /v2/{engine}/{project_id}/instances

表 5-6 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。

请求参数

表 5-7 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	String	实例名称。 由英文字符开头，只能由英文字母、数字、中划线、下划线组成，长度为4~64的字符。
description	否	String	实例的描述信息。 长度不超过1024的字符串。 **说明** \与"在json报文中属于特殊字符，如果参数值中需要显示\或者"字符，请在字符前增加转义字符\，比如\或者"。
engine	是	String	消息引擎。取值填写为：reliability。
engine_version	是	String	消息引擎的版本。取值填写为：4.8.0。
storage_space	是	Integer	存储空间。
vpc_id	是	String	虚拟私有云ID。 获取方法如下：登录虚拟私有云服务的控制台界面，在虚拟私有云的详情页面查找VPC ID。
subnet_id	是	String	子网信息。 获取方法如下：登录虚拟私有云服务的控制台界面，单击VPC下的子网，进入子网详情页面，查找网络ID。
security_group_id	是	String	指定实例所属的安全组。 获取方法如下：登录虚拟私有云服务的控制台界面，在安全组的详情页面查找安全组ID。

参数	是否必选	参数类型	描述
available_zones	是	Array of strings	创建节点到指定且有资源的可用区ID。请参考**查询可用区信息**获取可用区ID。 该参数不能为空数组或者数组的值为空， 请注意查看该可用区是否有资源。 创建RocketMQ实例，支持节点部署在1个或3个及3个以上的可用区。在为节点指定可用区时，用逗号分隔开。
product_id	是	String	RocketMQ实例规格。 • c6.4u8g.cluster.small：单个代理最大Topic数2000，单个代理最大消费组数2000 • c6.4u8g.cluster：单个代理最大Topic数4000，单个代理最大消费组数4000 • c6.8u16g.cluster：单个代理最大Topic数8000，单个代理最大消费组数8000 • c6.12u24g.cluster：单个代理最大Topic数12000，单个代理最大消费组数12000 • c6.16u32g.cluster：单个代理最大Topic数16000，单个代理最大消费组数16000
ssl_enable	否	Boolean	是否打开SSL加密访问。 • true：打开SSL加密访问。 • false：不打开SSL加密访问。
storage_spec_code	是	String	存储IO规格。 • dms.physical.storage.high.v2: 高IO类型磁盘 • dms.physical.storage.ultra.v2: 超高IO类型磁盘
enterprise_project_id	否	String	企业项目ID。若为企业项目账号，该参数必填。
enable_acl	否	Boolean	是否开启访问控制列表。
ipv6_enable	否	Boolean	是否支持IPv6。 • true: 支持 • false：不支持 缺省值：**false**

参数	是否必选	参数类型	描述
enable_publicip	否	Boolean	是否开启公网访问功能。默认不开启公网。 ● true：开启 ● false：不开启 缺省值：**false**
publicip_id	否	String	实例绑定的弹性IP地址的ID。 以英文逗号隔开多个弹性IP地址的ID。 如果开启了公网访问功能（即enable_publicip为true），该字段为必选。
broker_num	是	Integer	代理个数
bss_param	否	**BssParam** object	表示包周期计费模式的相关参数。 如果为空，则默认计费模式为按需计费；否则是包周期方式。

表 5-8 BssParam

参数	是否必选	参数类型	描述
is_auto_renew	否	Boolean	是否自动续订。 取值范围： ● true: 自动续订。 ● false: 不自动续订。 默认不自动续订。
charging_mode	否	String	计费模式。 功能说明：付费方式。 取值范围： ● prePaid：预付费，即包年包月； ● postPaid：后付费，即按需付费； 默认为postPaid。

参数	是否必选	参数类型	描述
is_auto_pay	否	Boolean	下单订购后，是否自动从客户的账户中支付，而不需要客户手动去进行支付。 取值范围： ● true：是（自动支付） ● false：否（需要客户手动支付） 默认为手动支付。
period_type	否	String	订购周期类型。 取值范围： ● month：月 ● year：年 **chargingMode为prePaid时生效且为必选值。**
period_num	否	Integer	订购周期数。 取值范围： ● periodType=month（周期类型为月）时，取值为[1，9]； ● periodType=year（周期类型为年）时，取值为[1，3]； **chargingMode为prePaid时生效且为必选值。**

响应参数

状态码： 200

表 5-9 响应 Body 参数

参数	参数类型	描述
instance_id	String	实例ID。

请求示例

创建一个RocketMQ实例，规格为4U8G*1，600GB的存储空间，按需付费。

```
POST https://{endpoint}/v2/reliability/{project_id}/instances

{
  "name" : "reliability-1751840557",
  "description" : "",
  "engine" : "reliability",
```

```
  "engine_version" : "4.8.0",
  "storage_space" : 600,
  "vpc_id" : "3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf",
  "subnet_id" : "0a0f1fcb-f019-458d-b9e5-301867394d50",
  "security_group_id" : "23c5977f-ff33-4b95-a73e-08d8a0bc4e6c",
  "available_zones" : [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
  "product_id" : "c6.4u8g.cluster",
  "enterprise_project_id" : "0",
  "ssl_enable" : false,
  "storage_spec_code" : "dms.physical.storage.ultra.v2",
  "ipv6_enable" : false,
  "enable_publicip" : false,
  "publicip_id" : "",
  "broker_num" : 1
}
```

响应示例

状态码： 200

创建实例成功。

```
{
  "instance_id" : "8959ab1c-7n1a-yyb1-a05t-93dfc361b32d"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个RocketMQ实例，规格为4U8G*1，600GB的存储空间，按需付费。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateInstanceByEngineSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
```

```
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateInstanceByEngineRequest request = new CreateInstanceByEngineRequest();
        request.withEngine(CreateInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
        CreateInstanceByEngineReq body = new CreateInstanceByEngineReq();
        List<String> listbodyAvailableZones = new ArrayList<>();
        listbodyAvailableZones.add("9f1c5806706d4c1fb0eb72f0a9b18c77");
        body.withBrokerNum(1);
        body.withPublicipId("");
        body.withEnablePublicip(false);
        body.withIpv6Enable(false);
        body.withEnterpriseProjectId("0");

body.withStorageSpecCode(CreateInstanceByEngineReq.StorageSpecCodeEnum.fromValue("dms.physical.stor
age.ultra.v2"));
        body.withSslEnable(false);
        body.withProductId(CreateInstanceByEngineReq.ProductIdEnum.fromValue("c6.4u8g.cluster"));
        body.withAvailableZones(listbodyAvailableZones);
        body.withSecurityGroupId("23c5977f-ff33-4b95-a73e-08d8a0bc4e6c");
        body.withSubnetId("0a0f1fcb-f019-458d-b9e5-301867394d50");
        body.withVpcId("3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf");
        body.withStorageSpace(600);
        body.withEngineVersion(CreateInstanceByEngineReq.EngineVersionEnum.fromValue("4.8.0"));
        body.withEngine(CreateInstanceByEngineReq.EngineEnum.fromValue("reliability"));
        body.withDescription("");
        body.withName("reliability-1751840557");
        request.withBody(body);
        try {
            CreateInstanceByEngineResponse response = client.createInstanceByEngine(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建一个RocketMQ实例，规格为4U8G*1，600GB的存储空间，按需付费。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
```

```
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateInstanceByEngineRequest()
        request.engine = "{engine}"
        listAvailableZonesbody = [
            "9f1c5806706d4c1fb0eb72f0a9b18c77"
        ]
        request.body = CreateInstanceByEngineReq(
            broker_num=1,
            publicip_id="",
            enable_publicip=False,
            ipv6_enable=False,
            enterprise_project_id="0",
            storage_spec_code="dms.physical.storage.ultra.v2",
            ssl_enable=False,
            product_id="c6.4u8g.cluster",
            available_zones=listAvailableZonesbody,
            security_group_id="23c5977f-ff33-4b95-a73e-08d8a0bc4e6c",
            subnet_id="0a0f1fcb-f019-458d-b9e5-301867394d50",
            vpc_id="3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf",
            storage_space=600,
            engine_version="4.8.0",
            engine="reliability",
            description="",
            name="reliability-1751840557"
        )
        response = client.create_instance_by_engine(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建一个RocketMQ实例，规格为4U8G*1，600GB的存储空间，按需付费。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
```

```
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateInstanceByEngineRequest{}
    request.Engine = model.GetCreateInstanceByEngineRequestEngineEnum().ENGINE
    var listAvailableZonesbody = []string{
        "9f1c5806706d4c1fb0eb72f0a9b18c77",
    }
    publicipIdCreateInstanceByEngineReq:= ""
    enablePublicipCreateInstanceByEngineReq:= false
    ipv6EnableCreateInstanceByEngineReq:= false
    enterpriseProjectIdCreateInstanceByEngineReq:= "0"
    sslEnableCreateInstanceByEngineReq:= false
    descriptionCreateInstanceByEngineReq:= ""
    request.Body = &model.CreateInstanceByEngineReq{
        BrokerNum: int32(1),
        PublicipId: &publicipIdCreateInstanceByEngineReq,
        EnablePublicip: &enablePublicipCreateInstanceByEngineReq,
        Ipv6Enable: &ipv6EnableCreateInstanceByEngineReq,
        EnterpriseProjectId: &enterpriseProjectIdCreateInstanceByEngineReq,
        StorageSpecCode:
model.GetCreateInstanceByEngineReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_ULTRA,
        SslEnable: &sslEnableCreateInstanceByEngineReq,
        ProductId: model.GetCreateInstanceByEngineReqProductIdEnum().C6_4U8G_CLUSTER,
        AvailableZones: listAvailableZonesbody,
        SecurityGroupId: "23c5977f-ff33-4b95-a73e-08d8a0bc4e6c",
        SubnetId: "0a0f1fcb-f019-458d-b9e5-301867394d50",
        VpcId: "3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf",
        StorageSpace: int32(600),
        EngineVersion: model.GetCreateInstanceByEngineReqEngineVersionEnum().E_4_8_0,
        Engine: model.GetCreateInstanceByEngineReqEngineEnum().RELIABILITY,
        Description: &descriptionCreateInstanceByEngineReq,
        Name: "reliability-1751840557",
    }
    response, err := client.CreateInstanceByEngine(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	创建实例成功。

错误码

请参见**错误码**。

5.1.3 查询指定实例

功能介绍

查询指定实例的详细信息。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}

表 5-10 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码： 200

表 5-11 响应 Body 参数

参数	参数类型	描述
name	String	实例名称。
engine	String	引擎。
status	String	状态。
description	String	消息描述。
type	String	实例类型：集群，cluster。
specification	String	实例规格。
engine_version	String	版本。
instance_id	String	实例ID。

参数	参数类型	描述
charging_mode	Integer	付费模式，1表示按需计费，0表示包年/包月计费。
vpc_id	String	私有云ID。
vpc_name	String	私有云名称。
created_at	String	完成创建时间。 格式为时间戳，指从格林威治时间1970年01月01日00时00分00秒起至指定时间的偏差总毫秒数。
product_id	String	产品标识。
security_group_id	String	安全组ID。
security_group_name	String	租户安全组名称。
subnet_id	String	子网ID。
subnet_name	String	子网名称。
subnet_cidr	String	子网路由（仅RocketMQ 5.x版本会显示此字段）。
available_zones	Array of strings	可用区ID列表。
available_zone_names	Array of strings	可用区名称列表。
user_id	String	用户ID。
user_name	String	用户名。
maintain_begin	String	维护时间窗开始时间，格式为HH:mm:ss。
maintain_end	String	维护时间窗结束时间，格式为HH:mm:ss。
enable_log_collection	Boolean	是否开启消息收集功能。
storage_space	Integer	存储空间，单位：GB。
used_storage_space	Integer	已用消息存储空间，单位：GB。
enable_publicip	Boolean	是否开启公网。
publicip_id	String	实例绑定的弹性IP地址的ID。以英文逗号隔开多个弹性IP地址的ID。如果开启了公网访问功能（即enable_publicip为true），该字段为必选。

参数	参数类型	描述
publicip_address	String	公网IP地址。
ssl_enable	Boolean	是否开启SSL。
cross_vpc_info	String	跨VPC访问信息。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	存储规格代码。
service_type	String	服务类型。
storage_type	String	存储类型。
extend_times	Long	扩展时间。
ipv6_enable	Boolean	是否开启IPv6。
support_features	String	实例支持的特性功能。
disk_encrypted	Boolean	是否开启磁盘加密。
ces_version	String	云监控版本。
node_num	Integer	节点数。
new_spec_billing_enable	Boolean	是否启用新规格计费。
enable_acl	Boolean	是否开启访问控制列表。
broker_num	Integer	节点数（仅RocketMQ 4.8.0版本会显示此字段）。
namesrv_address	String	元数据地址。
broker_address	String	业务数据地址。
public_namesrv_address	String	公网元数据地址。
public_broker_address	String	公网业务数据地址。
grpc_address	String	grpc连接地址（仅RocketMQ 5.x版本会显示此字段）。
public_grpc_address	String	公网grpc连接地址（仅RocketMQ 5.x版本会显示此字段）。

参数	参数类型	描述
enterprise_project_id	String	企业项目ID。
tags	Array of TagEntity objects	标签列表。
total_storage_space	Integer	总存储空间。
resource_spec_code	String	资源规格。

表 5-12 TagEntity

参数	参数类型	描述
key	String	标签键。 • 不能为空。 • 对于同一个实例，Key值唯一。 • 长度为1~128个字符（中文也可以输入128个字符）。 • 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ • 首尾字符不能为空格。
value	String	标签值。 • 长度为0~255个字符（中文也可以输入255个字符）。 • 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ • 首尾字符不能为空格。

请求示例

查询指定实例的详细信息。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}
```

响应示例

状态码： 200

查询实例成功。

- ```
  {
    "name" : "reliability-test",
    "engine" : "reliability",
  ```
  ```

```
 "status" : "RUNNING",
 "description" : "",
 "type" : "cluster",
 "specification" : "c6.4u8g.cluster * 1 broker",
 "engine_version" : "4.8.0",
 "instance_id" : "68fdc9a8-805e-439d-8dd9-25adc1c58bf6",
 "resource_spec_code" : "",
 "charging_mode" : 1,
 "vpc_id" : "3db8490c-4d6d-4d8f-8d3f-047b0de4c5cf",
 "vpc_name" : "vpc-1101840",
 "created_at" : "1636699753874",
 "product_id" : "c6.4u8g.cluster",
 "security_group_id" : "23c5977f-ff33-4b95-a73e-08d8a0bc4e6c",
 "security_group_name" : "Sys-default",
 "subnet_id" : "0a0f1fcb-f019-458d-b9e5-301867394d50",
 "subnet_name" : "subnet-boce",
 "subnet_cidr" : "192.168.1.0/24",
 "available_zones" : [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
 "available_zone_names" : [ "AZ3" ],
 "user_id" : "0b01fbb53600d4671fa8c00673c71260",
 "user_name" : "",
 "maintain_begin" : "02:00:00",
 "maintain_end" : "06:00:00",
 "enable_log_collection" : false,
 "storage_space" : 558,
 "total_storage_space" : 600,
 "used_storage_space" : 28,
 "enable_publicip" : true,
 "publicip_id" : "7e6b7beb-ef13-4805-878d-285890b17a12,87827295-84e1-4118-9ab3-
a90dd8c9dace,91306a18-5781-4529-b739-5aeec10ec4e6,a68e8f0e-
a24f-4521-910e-39696a56fcdf,bb1cd147-a7af-46ca-9548-99fec4034ba2",
 "publicip_address" : "***",
 "ssl_enable" : false,
 "cross_vpc_info" : "{\"192.168.1.21\":{\"advertised_ip\":\"192.168.1.21\",\"broker_port\":\"-\",\"port_id
\":\"fa020857-d899-497c-a5f8-8dd90ed67ff7\",\"namesrv_port\":\"8301\"},\"192.168.1.246\":
{\"advertised_ip\":\"192.168.1.246\",\"broker_port\":\"10100\",\"port_id\":\"c0f0586f-a4ee-41b5-a7f1-
b4e554bbf12d\",\"namesrv_port\":\"-\"},\"192.168.1.77\":{\"advertised_ip
\":\"192.168.1.77\",\"broker_port\":\"10101\",\"port_id\":\"8eb1d144-4315-402f-8498-37c9a10b630c
\",\"namesrv_port\":\"-\"},\"192.168.1.159\":{\"advertised_ip\":\"192.168.1.159\",\"broker_port\":\"-
\",\"port_id\":\"d35b22af-a202-4329-bda4-26e1bdf2aa8e\",\"namesrv_port
\":\"8300\"},\"192.168.1.14\":{\"advertised_ip\":\"192.168.1.14\",\"broker_port\":\"10102\",\"port_id
\":\"5fcb50fd-3af5-4123-a455-a67f1b926026\",\"namesrv_port\":\"-\"}}",
 "storage_resource_id" : "164bdaef-2e67-4fd5-be8b-a18f91d455a2",
 "storage_spec_code" : "dms.physical.storage.ultra.v2",
 "service_type" : "advanced",
 "storage_type" : "hec",
 "enterprise_project_id" : "0",
 "extend_times" : 0,
 "ipv6_enable" : false,
 "support_features" :
"kafka.crossvpc.domain.enable,feature.physerver.kafka.topic.accesspolicy,rabbitmq.plugin.management,
rocketmq.acl,roma_app_enable,auto_topic_switch,feature.physerver.kafka.user.manager,enable.new.aut
hinfo,route,kafka.config.dynamic.modify.enable,kafka.new.pod.port,feature.physerver.kafka.topic.modify
,message_trace_enable,features.pod.token.access,kafka.ssl.cert.modify.enable,roma.user.manage.no.sup
port,feature.physerver.kafka.pulbic.dynamic,features.log.collection,kafka.config.static.modify.enable",
 "disk_encrypted" : false,
 "ces_version" : "linux,v1,v2",
 "node_num" : 5,
 "new_spec_billing_enable" : true,
 "enable_acl" : false,
 "broker_num" : 1,
 "namesrv_address" : "***",
 "broker_address" : "***",
 "public_namesrv_address" : "***",
 "public_broker_address" : "***",
 "grpc_address" : "***",
 "public_grpc_address" : "***"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowInstanceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowInstanceRequest request = new ShowInstanceRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowInstanceResponse response = client.showInstance(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
```

```
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowInstanceRequest()
        request.instance_id = "{instance_id}"
        response = client.show_instance(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowInstanceRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```

```
  }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询实例成功。

错误码

请参见**错误码**。

5.1.4 删除指定的实例

功能介绍

删除指定的实例，释放该实例的所有资源。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}

表 5-13 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

无

响应参数

无

请求示例

删除指定的实例。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class DeleteInstanceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteInstanceRequest request = new DeleteInstanceRequest();
        request.withInstanceId("{instance_id}");
        try {
            DeleteInstanceResponse response = client.deleteInstance(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteInstanceRequest()
        request.instance_id = "{instance_id}"
        response = client.delete_instance(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
```

```
        Build())

    request := &model.DeleteInstanceRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.DeleteInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	删除指定的实例成功。

错误码

请参见**错误码**。

5.1.5 修改实例信息

功能介绍

修改实例的名称和描述信息。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}

表 5-14 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-15 请求 Body 参数

参数	是否必选	参数类型	描述
name	否	String	实例名称。 由英文字符开头，只能由英文字母、数字、中划线组成，长度为4~64的字符。
description	否	String	实例的描述信息。 长度不超过1024的字符串。 **说明** \与"在json报文中属于特殊字符，如果参数值中需要显示\或者"字符，请在字符前增加转义字符\，比如\或者"。
security_group_id	否	String	安全组ID。 获取方法如下：登录虚拟私有云服务的控制台界面，在安全组的详情页面查找安全组ID。
enable_acl	否	Boolean	ACL访问控制。
enable_publicip	否	Boolean	是否开启公网。
publicip_id	否	String	实例绑定的弹性IP地址的ID。 以英文逗号隔开多个弹性IP地址的ID。 如果开启了公网访问功能（即enable_publicip为true），该字段为必选。

响应参数

无

请求示例

修改RocketMQ实例的名称和描述信息。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}

{
  "name" : "rocketmq001",
  "description" : "RocketMQ description"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改RocketMQ实例的名称和描述信息。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class UpdateInstanceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withDescription("RocketMQ description");
        body.withName("rocketmq001");
        request.withBody(body);
        try {
            UpdateInstanceResponse response = client.updateInstance(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改RocketMQ实例的名称和描述信息。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
            description="RocketMQ description",
            name="rocketmq001"
        )
        response = client.update_instance(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改RocketMQ实例的名称和描述信息。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()
```

```
    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceRequest{}
    request.InstanceId = "{instance_id}"
    descriptionUpdateInstanceReq:= "RocketMQ description"
    nameUpdateInstanceReq:= "rocketmq001"
    request.Body = &model.UpdateInstanceReq{
        Description: &descriptionUpdateInstanceReq,
        Name: &nameUpdateInstanceReq,
    }
    response, err := client.UpdateInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	修改实例信息成功。

错误码

请参见**错误码**。

5.1.6 批量删除实例

功能介绍

批量删除实例。**实例删除后，实例中原有的数据将被删除，且没有备份，请谨慎操作。**

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/action

表 5-16 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。

请求参数

表 5-17 请求 Body 参数

参数	是否必选	参数类型	描述
instances	否	Array of strings	实例的ID列表。
action	是	String	对实例的操作：delete
all_failure	否	String	参数值为reliability，表示删除租户所有创建失败的RocketMQ实例。

响应参数

状态码： 200

表 5-18 响应 Body 参数

参数	参数类型	描述
results	Array of **results** objects	修改实例的结果。

表 5-19 results

参数	参数类型	描述
result	String	操作结果。
instance	String	实例ID。

请求示例

- 批量删除RocketMQ实例。

```
POST https://{endpoint}/v2/{project_id}/instances/action

{
```

```
 "action" : "delete",
 "instances" : [ "54602a9d-5e22-4239-9123-77e350df4a34", "7166cdea-
dbad-4d79-9610-7163e6f8b640" ]
}
```

- 删除所有创建失败的RocketMQ实例。

```
POST https://{endpoint}/v2/{project_id}/instances/action

{
 "action" : "delete",
 "all_failure" : "reliability"
}
```

响应示例

状态码： 200

批量删除实例成功。

```
{
  "results" : [ {
    "result" : "success",
    "instance" : "019cacb7-4ff0-4d3c-9f33-f5f7b7fdc0e6"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 批量删除RocketMQ实例。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class BatchDeleteInstancesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
```

```
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
        BatchDeleteInstancesRequest request = new BatchDeleteInstancesRequest();
        BatchDeleteInstanceReq body = new BatchDeleteInstanceReq();
        List<String> listbodyInstances = new ArrayList<>();
        listbodyInstances.add("54602a9d-5e22-4239-9123-77e350df4a34");
        listbodyInstances.add("7166cdea-dbad-4d79-9610-7163e6f8b640");
        body.withAction(BatchDeleteInstanceReq.ActionEnum.fromValue("delete"));
        body.withInstances(listbodyInstances);
        request.withBody(body);
        try {
            BatchDeleteInstancesResponse response = client.batchDeleteInstances(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 删除所有创建失败的RocketMQ实例。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class BatchDeleteInstancesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchDeleteInstancesRequest request = new BatchDeleteInstancesRequest();
        BatchDeleteInstanceReq body = new BatchDeleteInstanceReq();
        body.withAllFailure(BatchDeleteInstanceReq.AllFailureEnum.fromValue("reliability"));
        body.withAction(BatchDeleteInstanceReq.ActionEnum.fromValue("delete"));
        request.withBody(body);
        try {
            BatchDeleteInstancesResponse response = client.batchDeleteInstances(request);
```

```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 批量删除RocketMQ实例。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchDeleteInstancesRequest()
        listInstancesbody = [
            "54602a9d-5e22-4239-9123-77e350df4a34",
            "7166cdea-dbad-4d79-9610-7163e6f8b640"
        ]
        request.body = BatchDeleteInstanceReq(
            action="delete",
            instances=listInstancesbody
        )
        response = client.batch_delete_instances(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 删除所有创建失败的RocketMQ实例。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchDeleteInstancesRequest()
        request.body = BatchDeleteInstanceReq(
            all_failure="reliability",
            action="delete"
        )
        response = client.batch_delete_instances(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 批量删除RocketMQ实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())
```

```
    request := &model.BatchDeleteInstancesRequest{}
    var listInstancesbody = []string{
        "54602a9d-5e22-4239-9123-77e350df4a34",
        "7166cdea-dbad-4d79-9610-7163e6f8b640",
    }
    request.Body = &model.BatchDeleteInstanceReq{
        Action: model.GetBatchDeleteInstanceReqActionEnum().DELETE,
        Instances: &listInstancesbody,
    }
    response, err := client.BatchDeleteInstances(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 删除所有创建失败的RocketMQ实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchDeleteInstancesRequest{}
    allFailureBatchDeleteInstanceReq:= model.GetBatchDeleteInstanceReqAllFailureEnum().RELIABILITY
    request.Body = &model.BatchDeleteInstanceReq{
        AllFailure: &allFailureBatchDeleteInstanceReq,
        Action: model.GetBatchDeleteInstanceReqActionEnum().DELETE,
    }
    response, err := client.BatchDeleteInstances(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	批量删除实例成功。
204	删除所有创建失败的RocketMQ实例成功。

错误码

请参见**错误码**。

5.1.7 查询代理列表

功能介绍

查询代理列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/brokers

表 5-20 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码： 200

表 5-21 响应 Body 参数

参数	参数类型	描述
brokers	Array of **brokers** objects	代理列表。

表 5-22 brokers

参数	参数类型	描述
ids	Array of numbers	全部代理ID。
broker_name	String	节点名称。

请求示例

查询RocketMQ实例的代理列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/brokers
```

响应示例

状态码： 200

查询RocketMQ代理列表成功。

```
{
  "brokers" : [ {
    "ids" : [ 0, 1, 2 ],
    "broker_name" : "broker-0"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListBrokersSolution {

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

```
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListBrokersRequest request = new ListBrokersRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListBrokersResponse response = client.listBrokers(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListBrokersRequest()
        request.instance_id = "{instance_id}"
        response = client.list_brokers(request)
        print(response)
    except exceptions.ClientRequestException as e:
```

```
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListBrokersRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListBrokers(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询RocketMQ代理列表成功。

错误码

请参见**错误码**。

5.2 消费组管理

5.2.1 查询消费组列表

功能介绍

查询消费组列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/groups

表 5-23 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-24 Query 参数

参数	是否必选	参数类型	描述
group	否	String	消费组名称。
limit	否	Integer	查询数量。 缺省值：**10**
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。 缺省值：**0**

请求参数

无

响应参数

状态码： 200

表 5-25 响应 Body 参数

参数	参数类型	描述
total	Number	消费组总数。
groups	Array of **ConsumerGroup** objects	消费组列表。
max	Integer	最大可创建消费组数量。
remaining	Integer	剩余可创建消费组数量。
next_offset	Integer	下个分页的offset。
previous_offset	Integer	上个分页的offset。

表 5-26 ConsumerGroup

参数	参数类型	描述
enabled	Boolean	是否可以消费。
broadcast	Boolean	是否广播。
brokers	Array of strings	关联的代理列表。
name	String	消费组名称，只能由英文字母、数字、百分号、竖线、中划线、下划线组成，长度3~64个字符。
group_desc	String	消费组描述，长度0~200个字符。 最小长度：**0** 最大长度：**200**
retry_max_time	Integer	最大重试次数，取值范围为1~16。
createdAt	Long	创建时间戳。
permissions	Array of strings	权限集。
consume_orderly	Boolean	是否按序消费。

请求示例

查询RocketMQ实例的消费组列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups
```

响应示例

状态码： 200

查询消费组成功。

- ```
  {
    "total" : 1,
    "groups" : [ {
      "name" : "group-1",
      "enabled" : true,
      "broadcast" : false,
      "brokers" : [ "broker-0" ],
      "createdAt" : 1709087952686,
      "permissions" : [ ],
      "retry_max_time" : 16,
      "consume_orderly" : false
    } ],
    "max" : 4000,
    "remaining" : 3999,
    "next_offset" : -1,
    "previous_offset" : -1
  }
  ```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListInstanceConsumerGroupsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListInstanceConsumerGroupsRequest request = new ListInstanceConsumerGroupsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListInstanceConsumerGroupsResponse response = client.listInstanceConsumerGroups(request);
```
```

```
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListInstanceConsumerGroupsRequest()
        request.instance_id = "{instance_id}"
        response = client.list_instance_consumer_groups(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
```

```
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListInstanceConsumerGroupsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListInstanceConsumerGroups(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询消费组成功。

错误码

请参见**错误码**。

5.2.2 创建消费组或批量删除消费组

功能介绍

创建消费组或批量删除消费组。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/groups

表 5-27 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-28 Query 参数

参数	是否必选	参数类型	描述
action	否	String	批量删除消费组时使用，不配置则为创建接口。删除操作：delete。

请求参数

表 5-29 请求 Body 参数

参数	是否必选	参数类型	描述
groups	否	Array of strings	待删除的消费组列表。
name	否	String	消费组名称，只能由英文字母、数字、百分号、竖线、中划线、下划线组成，长度3~64个字符。
brokers	否	Array of strings	关联的代理列表（仅RocketMQ实例4.8.0版本需要填写此参数）。
broadcast	否	Boolean	是否广播。
retry_max_time	否	Integer	最大重试次数，取值范围为1~16。
enabled	否	Boolean	是否可以消费。
consume_orderly	否	Boolean	是否按序消费（仅RocketMQ实例5.x版本需要填写此参数）。
group_desc	否	String	消费组描述，长度0~200个字符。 最小长度：**0** 最大长度：**200**

响应参数

状态码： 200

表 5-30 响应 Body 参数

参数	参数类型	描述
job_id	String	删除消费组的任务ID
name	String	创建成功的消费组名称。

请求示例

- 4.8.0版本RocketMQ实例创建一个消费组，不允许以广播模式消费，关联代理为broker-0，最大重试次数为16。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups

{
  "name" : "consumer-group-test",
  "group_desc" : "group_description",
  "brokers" : [ "broker-0" ],
  "broadcast" : false,
  "retry_max_time" : 16
}
```

- 5.x版本RocketMQ实例创建一个消费组，不允许以广播模式消费，不按序消费，最大重试次数为16。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups

{
  "name" : "consumer-group-test",
  "group_desc" : "group_description",
  "consume_orderly" : false,
  "broadcast" : false,
  "retry_max_time" : 16
}
```

- 删除消费组consumer-group-test。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups?action=delete

{
  "groups" : [ "consumer-group-test" ]
}
```

响应示例

状态码： 200

创建消费组或批量删除消费组成功。

```
{
  "name" : "consumer-group-test"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 4.8.0版本RocketMQ实例创建一个消费组，不允许以广播模式消费，关联代理为broker-0，最大重试次数为16。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateConsumerGroupOrBatchDeleteConsumerGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConsumerGroupOrBatchDeleteConsumerGroupRequest request = new CreateConsumerGroupOrBatchDeleteConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        CreateConsumerGroupOrBatchDeleteConsumerGroupReq body = new CreateConsumerGroupOrBatchDeleteConsumerGroupReq();
        List<String> listbodyBrokers = new ArrayList<>();
        listbodyBrokers.add("broker-0");
        body.withGroupDesc("group_description");
        body.withName("consumer-group-test");
        body.withRetryMaxTime(16);
        body.withBrokers(listbodyBrokers);
        body.withBroadcast(false);
        request.withBody(body);
        try {
            CreateConsumerGroupOrBatchDeleteConsumerGroupResponse response = client.createConsumerGroupOrBatchDeleteConsumerGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
```

```
    }
}
```

- 5.x版本RocketMQ实例创建一个消费组，不允许以广播模式消费，不按序消费，最大重试次数为16。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class CreateConsumerGroupOrBatchDeleteConsumerGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConsumerGroupOrBatchDeleteConsumerGroupRequest request = new CreateConsumerGroupOrBatchDeleteConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        CreateConsumerGroupOrBatchDeleteConsumerGroupReq body = new CreateConsumerGroupOrBatchDeleteConsumerGroupReq();
        body.withConsumeOrderly(false);
        body.withGroupDesc("group_description");
        body.withName("consumer-group-test");
        body.withRetryMaxTime(16);
        body.withBroadcast(false);
        request.withBody(body);
        try {
            CreateConsumerGroupOrBatchDeleteConsumerGroupResponse response = client.createConsumerGroupOrBatchDeleteConsumerGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 删除消费组consumer-group-test。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateConsumerGroupOrBatchDeleteConsumerGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConsumerGroupOrBatchDeleteConsumerGroupRequest request = new CreateConsumerGroupOrBatchDeleteConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        CreateConsumerGroupOrBatchDeleteConsumerGroupReq body = new CreateConsumerGroupOrBatchDeleteConsumerGroupReq();
        List listbodyGroups = new ArrayList<>();
        listbodyGroups.add("consumer-group-test");
        body.withGroups(listbodyGroups);
        request.withBody(body);
        try {
            CreateConsumerGroupOrBatchDeleteConsumerGroupResponse response = client.createConsumerGroupOrBatchDeleteConsumerGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 4.8.0版本RocketMQ实例创建一个消费组，不允许以广播模式消费，关联代理为broker-0，最大重试次数为16。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateConsumerGroupOrBatchDeleteConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        listBrokersbody = [
            "broker-0"
        ]
        request.body = CreateConsumerGroupOrBatchDeleteConsumerGroupReq(
            group_desc="group_description",
            name="consumer-group-test",
            retry_max_time=16,
            brokers=listBrokersbody,
            broadcast=False
        )
        response = client.create_consumer_group_or_batch_delete_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 5.x版本RocketMQ实例创建一个消费组，不允许以广播模式消费，不按序消费，最大重试次数为16。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
```

```
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateConsumerGroupOrBatchDeleteConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateConsumerGroupOrBatchDeleteConsumerGroupReq(
            consume_orderly=False,
            group_desc="group_description",
            name="consumer-group-test",
            retry_max_time=16,
            broadcast=False
        )
        response = client.create_consumer_group_or_batch_delete_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 删除消费组consumer-group-test。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateConsumerGroupOrBatchDeleteConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        listGroupsbody = [
            "consumer-group-test"
        ]
        request.body = CreateConsumerGroupOrBatchDeleteConsumerGroupReq(
            groups=listGroupsbody
        )
        response = client.create_consumer_group_or_batch_delete_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 4.8.0版本RocketMQ实例创建一个消费组，不允许以广播模式消费，关联代理为broker-0，最大重试次数为16。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConsumerGroupOrBatchDeleteConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    var listBrokersbody = []string{
        "broker-0",
    }
    groupDescCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= "group_description"
    nameCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= "consumer-group-test"
    retryMaxTimeCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= int32(16)
    broadcastCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= false
    request.Body = &model.CreateConsumerGroupOrBatchDeleteConsumerGroupReq{
        GroupDesc: &groupDescCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        Name: &nameCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        RetryMaxTime: &retryMaxTimeCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        Brokers: &listBrokersbody,
        Broadcast: &broadcastCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
    }
    response, err := client.CreateConsumerGroupOrBatchDeleteConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 5.x版本RocketMQ实例创建一个消费组，不允许以广播模式消费，不按序消费，最大重试次数为16。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)
```

```
func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConsumerGroupOrBatchDeleteConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    consumeOrderlyCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= false
    groupDescCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= "group_description"
    nameCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= "consumer-group-test"
    retryMaxTimeCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= int32(16)
    broadcastCreateConsumerGroupOrBatchDeleteConsumerGroupReq:= false
    request.Body = &model.CreateConsumerGroupOrBatchDeleteConsumerGroupReq{
        ConsumeOrderly: &consumeOrderlyCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        GroupDesc: &groupDescCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        Name: &nameCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        RetryMaxTime: &retryMaxTimeCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
        Broadcast: &broadcastCreateConsumerGroupOrBatchDeleteConsumerGroupReq,
    }
    response, err := client.CreateConsumerGroupOrBatchDeleteConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 删除消费组consumer-group-test。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
```

```
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConsumerGroupOrBatchDeleteConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    var listGroupsbody = []string{
        "consumer-group-test",
    }
    request.Body = &model.CreateConsumerGroupOrBatchDeleteConsumerGroupReq{
        Groups: &listGroupsbody,
    }
    response, err := client.CreateConsumerGroupOrBatchDeleteConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	创建消费组或批量删除消费组成功。

错误码

请参见**错误码**。

5.2.3 批量修改消费组

功能介绍

批量修改消费组。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/groups

表 5-31 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-32 请求 Body 参数

参数	是否必选	参数类型	描述
groups	否	Array of **CreateOrUpdateConsumerGroup** objects	消费组列表。

表 5-33 CreateOrUpdateConsumerGroup

参数	是否必选	参数类型	描述
name	否	String	消费组名称，只能由英文字母、数字、百分号、竖线、中划线、下划线组成，长度3~64个字符。
brokers	否	Array of strings	关联的代理列表（仅RocketMQ实例4.8.0版本需要填写此参数）。
broadcast	否	Boolean	是否广播。
retry_max_time	否	Integer	最大重试次数，取值范围为1~16。
enabled	否	Boolean	是否可以消费。
consume_orderly	否	Boolean	是否按序消费（仅RocketMQ实例5.x版本需要填写此参数）。
group_desc	否	String	消费组描述，长度0~200个字符。 最小长度：**0** 最大长度：**200**

响应参数

状态码： 200

表 5-34 响应 Body 参数

参数	参数类型	描述
job_id	String	任务ID。

请求示例

批量修改消费组的参数，将consumer-group-test消费组的最大重试次数修改为16。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups

{
  "groups" : [ {
    "name" : "consumer-group-test",
    "enabled" : true,
    "broadcast" : false,
    "consume_orderly" : false,
    "retry_max_time" : 16
  } ]
}
```

响应示例

状态码： 200

批量修改消费组成功。

```
{
  "job_id" : "8abfa7b27da211df017da340427b0979"
}
```

SDK 代码示例

SDK代码示例如下。

Java

批量修改消费组的参数，将consumer-group-test消费组的最大重试次数修改为16。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class BatchUpdateConsumerGroupSolution {
```

```
    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchUpdateConsumerGroupRequest request = new BatchUpdateConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        BatchUpdateConsumerGroupReq body = new BatchUpdateConsumerGroupReq();
        List<CreateOrUpdateConsumerGroup> listbodyGroups = new ArrayList<>();
        listbodyGroups.add(
            new CreateOrUpdateConsumerGroup()
                .withName("consumer-group-test")
                .withBroadcast(false)
                .withRetryMaxTime(16)
                .withEnabled(true)
                .withConsumeOrderly(false)
        );
        body.withGroups(listbodyGroups);
        request.withBody(body);
        try {
            BatchUpdateConsumerGroupResponse response = client.batchUpdateConsumerGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

批量修改消费组的参数，将consumer-group-test消费组的最大重试次数修改为16。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
```

```
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchUpdateConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        listGroupsbody = [
            CreateOrUpdateConsumerGroup(
                name="consumer-group-test",
                broadcast=False,
                retry_max_time=16,
                enabled=True,
                consume_orderly=False
            )
        ]
        request.body = BatchUpdateConsumerGroupReq(
            groups=listGroupsbody
        )
        response = client.batch_update_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

批量修改消费组的参数，将consumer-group-test消费组的最大重试次数修改为16。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchUpdateConsumerGroupRequest{}
```

```
    request.InstanceId = "{instance_id}"
    nameGroups:= "consumer-group-test"
    broadcastGroups:= false
    retryMaxTimeGroups:= int32(16)
    enabledGroups:= true
    consumeOrderlyGroups:= false
    var listGroupsbody = []model.CreateOrUpdateConsumerGroup{
        {
            Name: &nameGroups,
            Broadcast: &broadcastGroups,
            RetryMaxTime: &retryMaxTimeGroups,
            Enabled: &enabledGroups,
            ConsumeOrderly: &consumeOrderlyGroups,
        },
    }
    request.Body = &model.BatchUpdateConsumerGroupReq{
        Groups: &listGroupsbody,
    }
    response, err := client.BatchUpdateConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	批量修改消费组成功。

错误码

请参见**错误码**。

5.2.4 删除指定消费组

功能介绍

删除指定消费组。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}/groups/{group}

表 5-35 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group	是	String	消费组名称。

请求参数

无

响应参数

无

请求示例

删除指定的消费组。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups/{group}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class DeleteConsumerGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";
```

```
        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteConsumerGroupRequest request = new DeleteConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        try {
            DeleteConsumerGroupResponse response = client.deleteConsumerGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        response = client.delete_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    response, err := client.DeleteConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	操作成功。

错误码

请参见**错误码**。

5.2.5 查询指定消费组

功能介绍

查询指定消费组详情。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/groups/{group}

表 5-36 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group	是	String	消费组名称。

请求参数

无

响应参数

状态码： 200

表 5-37 响应 Body 参数

参数	参数类型	描述
enabled	Boolean	是否可以消费。
broadcast	Boolean	是否广播。
brokers	Array of strings	关联的代理列表。
name	String	消费组名称。
group_desc	String	消费组描述。
retry_max_time	Integer	最大重试次数。
app_id	String	应用id。

参数	参数类型	描述
app_name	String	应用名称。
permissions	Array of strings	权限。

请求示例

查询指定消费组的详细信息。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups/{group}
```

响应示例

状态码： 200

查询消费组成功。

```
{
  "name" : "test",
  "enabled" : true,
  "broadcast" : true,
  "brokers" : [ "broker-0" ],
  "permissions" : [ ],
  "retry_max_time" : 10,
  "app_id" : null,
  "app_name" : null
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
```

```
            .withProjectId(projectId)
            .withAk(ak)
            .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowGroupRequest request = new ShowGroupRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        try {
            ShowGroupResponse response = client.showGroup(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowGroupRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        response = client.show_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowGroupRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    response, err := client.ShowGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询消费组成功。

错误码

请参见**错误码**。

5.2.6 修改消费组

功能介绍

修改指定消费组参数。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/groups/{group}

表 5-38 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group	是	String	消费组名称。

请求参数

表 5-39 请求 Body 参数

参数	是否必选	参数类型	描述
enabled	是	Boolean	是否可以消费。
broadcast	是	Boolean	是否广播。
brokers	否	Array of strings	关联的代理列表。
name	否	String	待修改参数的消费组（消费组名称不支持修改）。
retry_max_time	是	Integer	最大重试次数，取值范围为1~16。

响应参数

无

请求示例

修改consumer-group-test消费组的参数，最大重试次数修改为16。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups/{group}

{
  "name" : "consumer-group-test",
  "enabled" : true,
  "retry_max_time" : 16,
  "broadcast" : true
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改consumer-group-test消费组的参数，最大重试次数修改为16。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class UpdateConsumerGroupSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateConsumerGroupRequest request = new UpdateConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        UpdateConsumerGroup body = new UpdateConsumerGroup();
        body.withRetryMaxTime(16);
        body.withName("consumer-group-test");
        body.withBroadcast(true);
        body.withEnabled(true);
        request.withBody(body);
        try {
            UpdateConsumerGroupResponse response = client.updateConsumerGroup(request);
            System.out.println(response.toString());
```

```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改consumer-group-test消费组的参数，最大重试次数修改为16。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        request.body = UpdateConsumerGroup(
            retry_max_time=16,
            name="consumer-group-test",
            broadcast=True,
            enabled=True
        )
        response = client.update_consumer_group(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改consumer-group-test消费组的参数，最大重试次数修改为16。

```
package main

import (
    "fmt"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    nameUpdateConsumerGroup:= "consumer-group-test"
    request.Body = &model.UpdateConsumerGroup{
        RetryMaxTime: int32(16),
        Name: &nameUpdateConsumerGroup,
        Broadcast: true,
        Enabled: true,
    }
    response, err := client.UpdateConsumerGroup(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	操作成功。

错误码

请参见**错误码**。

5.2.7 查询消费列表或详情

功能介绍

查询消费列表或详情。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/groups/{group}/topics

表 5-40 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group	是	String	消费组名称。

表 5-41 Query 参数

参数	是否必选	参数类型	描述
topic	否	String	待查询的topic，不指定时查询topic列表，指定时查询详情。
limit	否	Integer	当次查询返回的最大个数，默认值为10，取值范围为1~50。 最小值：**1** 最大值：**50** 缺省值：**10**
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。 最小值：**0** 缺省值：**0**

请求参数

无

响应参数

状态码： 200

表 5-42 响应 Body 参数

参数	参数类型	描述
topics	Array of strings	Topic列表（当查询topic消费“列表”时才显示此参数）。
total	Integer	Topic总数（当查询topic消费“列表”时才显示此参数）。
lag	Long	消费堆积总数
max_offset	Long	消息总数
consumer_offset	Long	已消费消息数
brokers	Array of **Brokers** objects	Topic关联代理（当查询topic消费“详情”才显示此参数）。

表 5-43 Brokers

参数	参数类型	描述
broker_name	String	Topic关联代理名称
queues	Array of **Queue** objects	关联代理的队列详情

表 5-44 Queue

参数	参数类型	描述
id	Integer	队列ID
lag	Long	队列消费堆积总数
broker_offset	Long	队列消息总数
consumer_offset	Long	已消费消息数
last_message_time	Long	最新消费消息的存储时间，unix毫秒时间戳格式

请求示例

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups/{group}/topics?topic=test0001
```

响应示例

状态码： 200

查询消费列表或详情成功。

- 查询消费组的topic列表成功。

```
{
  "topics" : [ "topic-test" ],
  "total" : 1
}
```

- 查询消费组详情成功。

```
{
  "lag" : 0,
  "max_offset" : 1,
  "consumer_offset" : 1,
  "brokers" : [ {
    "broker_name" : "broker-0",
    "queues" : [ {
      "id" : 0,
      "lag" : 0,
      "broker_offset" : 0,
      "consumer_offset" : 0,
      "last_message_time" : 0
    }, {
      "id" : 1,
      "lag" : 0,
      "broker_offset" : 1,
      "consumer_offset" : 1,
      "last_message_time" : 1679398537088
    }, {
      "id" : 0,
      "lag" : 0,
      "broker_offset" : 0,
      "consumer_offset" : 0,
      "last_message_time" : 0
    } ]
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowConsumerListOrDetailsSolution {

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

```
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowConsumerListOrDetailsRequest request = new ShowConsumerListOrDetailsRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        try {
            ShowConsumerListOrDetailsResponse response = client.showConsumerListOrDetails(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowConsumerListOrDetailsRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        response = client.show_consumer_list_or_details(request)
```

```
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowConsumerListOrDetailsRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    response, err := client.ShowConsumerListOrDetails(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询消费列表或详情成功。

错误码

请参见**错误码**。

5.2.8 重置消费进度

功能介绍

重置消费进度。

调用方法

请参见**如何调用API**。

URI

POST /v2/{engine}/{project_id}/instances/{instance_id}/groups/{group_id}/reset-message-offset

表 5-45 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎类型：reliability。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group_id	是	String	消费组名称。

请求参数

表 5-46 请求 Body 参数

参数	是否必选	参数类型	描述
topic	是	String	重置的主题。
timestamp	是	String	重置的时间。

响应参数

状态码： 200

表 5-47 响应 Body 参数

参数	参数类型	描述
queues	Array of **queues** objects	重置的队列。

表 5-48 queues

参数	参数类型	描述
broker_name	String	队列所在的broker。
queue_id	Integer	队列ID。
timestamp_offset	Long	重置消费进度。

请求示例

重置topic_01主题的消费进度到指定时间点。

```
POST https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/groups/{group_id}/reset-message-offset

{
  "topic" : "topic_01",
  "timestamp" : 1662652800000
}
```

响应示例

状态码： 200

重置消费进度成功。

```
{
  "queues" : [ {
    "broker_name" : "broker-1",
    "queue_id" : 0,
    "timestamp_offset" : 0
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

重置topic_01主题的消费进度到指定时间点。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
```

```
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ResetConsumeOffsetSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetConsumeOffsetRequest request = new ResetConsumeOffsetRequest();
        request.withEngine(ResetConsumeOffsetRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        request.withGroupId("{group_id}");
        ResetConsumeOffsetReq body = new ResetConsumeOffsetReq();
        body.withTimestamp("1662652800000");
        body.withTopic("topic_01");
        request.withBody(body);
        try {
            ResetConsumeOffsetResponse response = client.resetConsumeOffset(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

重置topic_01主题的消费进度到指定时间点。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
```

```
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResetConsumeOffsetRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.group_id = "{group_id}"
        request.body = ResetConsumeOffsetReq(
            timestamp="1662652800000",
            topic="topic_01"
        )
        response = client.reset_consume_offset(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

重置topic_01主题的消费进度到指定时间点。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetConsumeOffsetRequest{}
    request.Engine = model.GetResetConsumeOffsetRequestEngineEnum().ENGINE
```

```
    request.InstanceId = "{instance_id}"
    request.GroupId = "{group_id}"
    request.Body = &model.ResetConsumeOffsetReq{
        Timestamp: "1662652800000",
        Topic: "topic_01",
    }
    response, err := client.ResetConsumeOffset(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	重置消费进度成功。

错误码

请参见**错误码**。

5.2.9 查询消费者列表

功能介绍

查询消费组内消费者列表

调用方法

请参见**如何调用API**。

URI

GET /v2/rocketmq/{project_id}/instances/{instance_id}/groups/{group}/clients

表 5-49 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID
group	是	String	消费组名称

表 5-50 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	查询数量，取值范围为1~50。
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。
is_detail	否	Boolean	是否查询消费者详细列表，参数为“true”则表示查询详细列表，否则表示查询简易列表。

请求参数

无

响应参数

状态码： 200

表 5-51 响应 Body 参数

参数	参数类型	描述
group_name	String	消费组名称
online	Boolean	消费组是否在线
subscription_consistency	Boolean	订阅关系是否一致
total	Integer	消费者总数
next_offset	Integer	下个分页的offset
previous_offset	Integer	上个分页的offset
clients	Array of **ClientData** objects	消费者订阅详情列表

表 5-52 ClientData

参数	参数类型	描述
language	String	客户端语言
version	String	客户端版本
client_id	String	客户端ID

参数	参数类型	描述
client_addr	String	客户端地址
subscriptions	Array of **Subscription** objects	订阅关系列表

表 5-53 Subscription

参数	参数类型	描述
topic	String	订阅的topic名称
type	String	订阅类型，取值如下：TAG和SQL92
expression	String	订阅tag字符

请求示例

查询指定消费组的消费者列表，查询数量为10，从offset=0开始查询。

```
GET https://{endpoint}/v2/rocketmq/{project_id}/instances/{instance_id}/groups/{group}/clients?offset=0&limit=10&is_detail=true
```

响应示例

状态码： 200

查询消费者订阅详情成功。

```
{
  "group_name" : "test",
  "online" : true,
  "subscription_consistency" : true,
  "total" : 1,
  "next_offset" : -1,
  "previous_offset" : -1,
  "clients" : [ {
    "client_id" : "192.168.0.1@consumer1",
    "language" : "JAVA",
    "version" : "V4_8_0",
    "client_addr" : "192.168.0.1:65233",
    "subscriptions" : [ {
      "topic" : "topicA",
      "type" : "TAG",
      "expression" : "tagA"
    } ]
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowConsumerConnectionsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowConsumerConnectionsRequest request = new ShowConsumerConnectionsRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        try {
            ShowConsumerConnectionsResponse response = client.showConsumerConnections(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
```

```
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowConsumerConnectionsRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        response = client.show_consumer_connections(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowConsumerConnectionsRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    response, err := client.ShowConsumerConnections(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询消费者订阅详情成功。

错误码

请参见**错误码**。

5.3 Topic 管理

5.3.1 创建主题或批量删除主题

功能介绍

创建主题或批量删除主题。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/topics

表 5-54 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-55 Query 参数

参数	是否必选	参数类型	描述
action	否	String	批量删除topic时使用，不配置则为创建接口。删除操作：delete。

请求参数

表 5-56 请求 Body 参数

参数	是否必选	参数类型	描述
name	否	String	主题名称，只能由英文字母、数字、百分号、竖线、中划线、下划线组成，长度3~64个字符。
brokers	否	Array of strings	关联的代理（仅RocketMQ实例4.8.0版本需要填写此参数）。
queue_num	否	Number	队列数，范围1~50。
queues	否	Array of **queues** objects	队列（仅RocketMQ实例4.8.0版本需要填写此参数）。
permission	否	String	权限（仅RocketMQ实例4.8.0版本需要填写此参数）。 取值范围： • pub（发布） • sub（订阅） • all（发布+订阅） 缺省值：**all**
message_type	否	String	消息类型（仅RocketMQ实例5.x版本需要填写此参数）。 取值范围： • NORMAL（普通消息） • FIFO（顺序消息） • DELAY（定时消息） • TRANSACTION（事务消息）
topics	否	Array of strings	主题列表，当批量删除主题时使用。

表 5-57 queues

参数	是否必选	参数类型	描述
broker	否	String	关联的代理。
queue_num	否	Number	队列数，范围1~50。

响应参数

状态码： 200

表 5-58 响应 Body 参数

参数	参数类型	描述
id	String	主题名称。
job_id	String	删除主题任务ID。

请求示例

- 4.8.0版本RocketMQ实例创建一个主题，关联的代理为broker-0，队列数为3。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics

{
  "name" : "topic-test",
  "brokers" : [ "broker-0" ],
  "permission" : "all",
  "queues" : [ {
    "broker" : "broker-0",
    "queue_num" : 3
  } ]
}
```

- 5.x版本RocketMQ实例创建一个主题，消息类型为普通。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics

{
  "name" : "topic-test",
  "message_type" : "NORMAL"
}
```

响应示例

状态码： 200

创建主题或批量删除主题成功。

```
{
  "id" : "topic-test"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 4.8.0版本RocketMQ实例创建一个主题，关联的代理为broker-0，队列数为3。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
```

```
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateTopicOrBatchDeleteTopicSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateTopicOrBatchDeleteTopicRequest request = new CreateTopicOrBatchDeleteTopicRequest();
        request.withInstanceId("{instance_id}");
        CreateTopicOrBatchDeleteTopicReq body = new CreateTopicOrBatchDeleteTopicReq();
        List<CreateTopicReqQueues> listbodyQueues = new ArrayList<>();
        listbodyQueues.add(
            new CreateTopicReqQueues()
                .withBroker("broker-0")
                .withQueueNum(java.math.BigDecimal.valueOf(3))
        );
        List<String> listbodyBrokers = new ArrayList<>();
        listbodyBrokers.add("broker-0");
        body.withPermission(CreateTopicOrBatchDeleteTopicReq.PermissionEnum.fromValue("all"));
        body.withName("topic-test");
        body.withQueues(listbodyQueues);
        body.withBrokers(listbodyBrokers);
        request.withBody(body);
        try {
            CreateTopicOrBatchDeleteTopicResponse response = client.createTopicOrBatchDeleteTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 5.x版本RocketMQ实例创建一个主题，消息类型为普通。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class CreateTopicOrBatchDeleteTopicSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateTopicOrBatchDeleteTopicRequest request = new CreateTopicOrBatchDeleteTopicRequest();
        request.withInstanceId("{instance_id}");
        CreateTopicOrBatchDeleteTopicReq body = new CreateTopicOrBatchDeleteTopicReq();

body.withMessageType(CreateTopicOrBatchDeleteTopicReq.MessageTypeEnum.fromValue("NORMAL"));
        body.withName("topic-test");
        request.withBody(body);
        try {
            CreateTopicOrBatchDeleteTopicResponse response = client.createTopicOrBatchDeleteTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

- 4.8.0版本RocketMQ实例创建一个主题，关联的代理为broker-0，队列数为3。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
```

```
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateTopicOrBatchDeleteTopicRequest()
        request.instance_id = "{instance_id}"
        listQueuesbody = [
            CreateTopicReqQueues(
                broker="broker-0",
                queue_num=3
            )
        ]
        listBrokersbody = [
            "broker-0"
        ]
        request.body = CreateTopicOrBatchDeleteTopicReq(
            permission="all",
            name="topic-test",
            queues=listQueuesbody,
            brokers=listBrokersbody
        )
        response = client.create_topic_or_batch_delete_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 5.x版本RocketMQ实例创建一个主题，消息类型为普通。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateTopicOrBatchDeleteTopicRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateTopicOrBatchDeleteTopicReq(
            message_type="NORMAL",
```

```
            name="topic-test"
        )
        response = client.create_topic_or_batch_delete_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 4.8.0版本RocketMQ实例创建一个主题，关联的代理为broker-0，队列数为3。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateTopicOrBatchDeleteTopicRequest{}
    request.InstanceId = "{instance_id}"
    brokerQueues:= "broker-0"
    queueNumQueues:= float32(3)
    var listQueuesbody = []model.CreateTopicReqQueues{
        {
            Broker: &brokerQueues,
            QueueNum: &queueNumQueues,
        },
    }
    var listBrokersbody = []string{
        "broker-0",
    }
    permissionCreateTopicOrBatchDeleteTopicReq:= model.GetCreateTopicOrBatchDeleteTopicReqPermissionEnum().ALL
    nameCreateTopicOrBatchDeleteTopicReq:= "topic-test"
    request.Body = &model.CreateTopicOrBatchDeleteTopicReq{
        Permission: &permissionCreateTopicOrBatchDeleteTopicReq,
        Name: &nameCreateTopicOrBatchDeleteTopicReq,
        Queues: &listQueuesbody,
        Brokers: &listBrokersbody,
    }
    response, err := client.CreateTopicOrBatchDeleteTopic(request)
```

```
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 5.x版本RocketMQ实例创建一个主题，消息类型为普通。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateTopicOrBatchDeleteTopicRequest{}
    request.InstanceId = "{instance_id}"
    messageTypeCreateTopicOrBatchDeleteTopicReq:= model.GetCreateTopicOrBatchDeleteTopicReqMessageTypeEnum().NORMAL
    nameCreateTopicOrBatchDeleteTopicReq:= "topic-test"
    request.Body = &model.CreateTopicOrBatchDeleteTopicReq{
        MessageType: &messageTypeCreateTopicOrBatchDeleteTopicReq,
        Name: &nameCreateTopicOrBatchDeleteTopicReq,
    }
    response, err := client.CreateTopicOrBatchDeleteTopic(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	创建主题或批量删除主题成功。

错误码

请参见**错误码**。

5.3.2 查询主题列表

功能介绍

该接口用于查询指定RocketMQ实例的Topic列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/topics

表 5-59 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-60 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	查询数量，取值范围为1~50。
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。

请求参数

无

响应参数

状态码： 200

表 5-61 响应 Body 参数

参数	参数类型	描述
total	Integer	topic总数。
max	Integer	最大可创建topic数量。
remaining	Integer	剩余可创建topic数量。
next_offset	Integer	下个分页的offset。
previous_offset	Integer	上个分页的offset。
topics	Array of **Topic** objects	topic列表。

表 5-62 Topic

参数	参数类型	描述
name	String	topic名称。
total_read_queue_num	Number	总读队列个数。
total_write_queue_num	Number	总写队列个数。
permission	String	权限。
brokers	Array of **brokers** objects	关联的代理。
message_type	String	消息类型（RocketMQ实例5.x版本才包含此参数）。

表 5-63 brokers

参数	参数类型	描述
broker_name	String	代理名称。
read_queue_num	Number	读队列个数。
write_queue_num	Number	写队列个数。

请求示例

查询指定RocketMQ实例的主题列表，查询数量为10，从offset=0开始查询。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics?offset=0&limit=10
```

响应示例

状态码： 200

查询成功。

```
{
  "total" : "3,",
  "max" : "2000,",
  "remaining" : "1997,",
  "next_offset" : "-1,",
  "previous_offset" : "-1,",
  "topics" : [ {
    "name" : "topic-1",
    "total_read_queue_num" : 3,
    "total_write_queue_num" : 3,
    "permission" : "all",
    "brokers" : [ {
      "broker_name" : "broker-0",
      "read_queue_num" : 3,
      "write_queue_num" : 3
    } ],
    "message_type" : "NORMAL"
  }, {
    "name" : "topic-2",
    "total_read_queue_num" : 3,
    "total_write_queue_num" : 3,
    "permission" : "all",
    "brokers" : [ {
      "broker_name" : "broker-0",
      "read_queue_num" : 3,
      "write_queue_num" : 3
    } ],
    "message_type" : "NORMAL"
  }, {
    "name" : "topic-3",
    "total_read_queue_num" : 3,
    "total_write_queue_num" : 3,
    "permission" : "all",
    "brokers" : [ {
      "broker_name" : "broker-0",
      "read_queue_num" : 3,
      "write_queue_num" : 3
    } ],
    "message_type" : "NORMAL"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListRocketInstanceTopicsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListRocketInstanceTopicsRequest request = new ListRocketInstanceTopicsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListRocketInstanceTopicsResponse response = client.listRocketInstanceTopics(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
```

```
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListRocketInstanceTopicsRequest()
        request.instance_id = "{instance_id}"
        response = client.list_rocket_instance_topics(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListRocketInstanceTopicsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListRocketInstanceTopics(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询成功。

错误码

请参见**错误码**。

5.3.3 删除指定主题

功能介绍

删除指定主题。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}/topics/{topic}

表 5-64 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

请求参数

无

响应参数

无

请求示例

删除指定的主题。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class DeleteTopicSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteTopicRequest request = new DeleteTopicRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            DeleteTopicResponse response = client.deleteTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8
```

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteTopicRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.delete_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteTopicRequest{}
```

```
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.DeleteTopic(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	操作成功。

错误码

请参见**错误码**。

5.3.4 查询单个主题

功能介绍

查询单个主题。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/topics/{topic}

表 5-65 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

请求参数

无

响应参数

状态码： 200

表 5-66 响应 Body 参数

参数	参数类型	描述
name	String	topic名称。
total_read_queue_num	Number	总读队列个数。
total_write_queue_num	Number	总写队列个数。
permission	String	权限。
brokers	Array of **brokers** objects	关联的代理。
message_type	String	消息类型（RocketMQ实例5.x版本才包含此参数）。

表 5-67 brokers

参数	参数类型	描述
broker_name	String	代理名称。
read_queue_num	Number	读队列个数。
write_queue_num	Number	写队列个数。

请求示例

查询指定的单个主题的详细信息。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}
```

响应示例

状态码： 200

查询单个主题成功。

```
{
  "total_read_queue_num" : 3,
```

```
  "total_write_queue_num" : 3,
  "permission" : "all",
  "brokers" : [ {
    "broker_name" : "broker-0",
    "read_queue_num" : 3,
    "write_queue_num" : 3
  } ],
  "message_type" : "NORMAL"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowOneTopicSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowOneTopicRequest request = new ShowOneTopicRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ShowOneTopicResponse response = client.showOneTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowOneTopicRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.show_one_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
```

```
            WithCredential(auth).
            Build())

    request := &model.ShowOneTopicRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ShowOneTopic(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询单个主题成功。

错误码

请参见**错误码**。

5.3.5 修改主题

功能介绍

修改主题。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/topics/{topic}

表 5-68 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

请求参数

表 5-69 请求 Body 参数

参数	是否必选	参数类型	描述
read_queue_num	否	Number	总读队列个数。
write_queue_num	否	Number	总写队列个数。
permission	否	String	权限。

响应参数

无

请求示例

修改指定主题的参数，总读队列个数修改为3，总写队列个数修改为3。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}

{
  "read_queue_num" : 3,
  "write_queue_num" : 3,
  "permission" : "all"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改指定主题的参数，总读队列个数修改为3，总写队列个数修改为3。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class UpdateTopicSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
```

```
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateTopicRequest request = new UpdateTopicRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        UpdateTopicReq body = new UpdateTopicReq();
        body.withPermission(UpdateTopicReq.PermissionEnum.fromValue("all"));
        body.withWriteQueueNum(java.math.BigDecimal.valueOf(3));
        body.withReadQueueNum(java.math.BigDecimal.valueOf(3));
        request.withBody(body);
        try {
            UpdateTopicResponse response = client.updateTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改指定主题的参数，总读队列个数修改为3，总写队列个数修改为3。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()
```

```
    try:
        request = UpdateTopicRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        request.body = UpdateTopicReq(
            permission="all",
            write_queue_num=3,
            read_queue_num=3
        )
        response = client.update_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改指定主题的参数，总读队列个数修改为3，总写队列个数修改为3。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateTopicRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    permissionUpdateTopicReq:= model.GetUpdateTopicReqPermissionEnum().ALL
    writeQueueNumUpdateTopicReq:= float32(3)
    readQueueNumUpdateTopicReq:= float32(3)
    request.Body = &model.UpdateTopicReq{
        Permission: &permissionUpdateTopicReq,
        WriteQueueNum: &writeQueueNumUpdateTopicReq,
        ReadQueueNum: &readQueueNumUpdateTopicReq,
    }
    response, err := client.UpdateTopic(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
```

```
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	操作成功。

错误码

请参见**错误码**。

5.3.6 查询主题消费组列表

功能介绍

查询主题消费组列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/topics/{topic}/groups

表 5-70 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

表 5-71 Query 参数

参数	是否必选	参数类型	描述
limit	否	Integer	当次查询返回的最大个数，默认值为10，取值范围为1~50。

参数	是否必选	参数类型	描述
offset	否	Integer	偏移量，表示从此偏移量开始查询，offset大于等于0。

请求参数

无

响应参数

状态码： 200

表 5-72 响应 Body 参数

参数	参数类型	描述
groups	Array of strings	消费组列表。

请求示例

查询指定主题的消费组列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}/groups
```

响应示例

状态码： 200

查询主题消费组列表成功。

```
{
  "groups" : [ "CID_JODIE_1", "test_consumer" ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListConsumerGroupOfTopicSolution {
```

```
    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListConsumerGroupOfTopicRequest request = new ListConsumerGroupOfTopicRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ListConsumerGroupOfTopicResponse response = client.listConsumerGroupOfTopic(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListConsumerGroupOfTopicRequest()
        request.instance_id = "{instance_id}"
```

```
        request.topic = "{topic}"
        response = client.list_consumer_group_of_topic(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListConsumerGroupOfTopicRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ListConsumerGroupOfTopic(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询主题消费组列表成功。

错误码

请参见**错误码**。

5.3.7 查询主题的消息数

功能介绍

查询主题的消息数。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/topics/{topic}/status

表 5-73 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

请求参数

无

响应参数

状态码： 200

表 5-74 响应 Body 参数

参数	参数类型	描述
max_offset	Integer	最大偏移量。
min_offset	Integer	最小偏移量。
brokers	Array of **brokers** objects	代理。

表 5-75 brokers

参数	参数类型	描述
queues	Array of **queues** objects	队列列表。
broker_name	String	节点名称。

表 5-76 queues

参数	参数类型	描述
id	Integer	队列ID。
min_offset	Integer	最小偏移量。
max_offset	Integer	最大偏移量。
last_message_time	Long	最后一条消息的时间。

请求示例

查询指定主题的消息数。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}/status
```

响应示例

状态码： 200

查询主题的消息数成功。

- 查询主题的消息数成功。

```
{
  "brokers" : {
    "queues" : [ {
      "id" : 0,
      "min_offset" : 0,
      "max_offset" : 2,
      "last_message_time" : 1662689877152
    } ],
    "broker_name" : "broker-0"
  }
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;
```

```
import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowTopicStatusSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowTopicStatusRequest request = new ShowTopicStatusRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ShowTopicStatusResponse response = client.showTopicStatus(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"
```

```
    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowTopicStatusRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.show_topic_status(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowTopicStatusRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ShowTopicStatus(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询主题的消息数成功。

错误码

请参见**错误码**。

5.4 消息管理

5.4.1 查询消息

功能介绍

查询消息。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/{project_id}/instances/{instance_id}/messages

表 5-77 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-78 Query 参数

参数	是否必选	参数类型	描述
topic	是	String	主题名称。
limit	否	String	查询数量。
offset	否	String	偏移量，表示从此偏移量开始查询， offset大于等于0。
key	否	String	消息的key
start_time	否	String	开始时间（不通过msg_id精确查询消息时，此参数必填）。
end_time	否	String	结束时间（不通过msg_id精确查询消息时，此参数必填）。
msg_id	否	String	消息ID。

请求参数

无

响应参数

状态码： 200

表 5-79 响应 Body 参数

参数	参数类型	描述
messages	Array of **Message** objects	消息列表。
total	Number	消息总数。

表 5-80 Message

参数	参数类型	描述
msg_id	String	消息ID。
instance_id	String	实例ID。
topic	String	主题名称。
store_timestamp	Number	存储消息的时间。

参数	参数类型	描述
born_timestamp	Number	产生消息的时间。
reconsume_times	Integer	重试次数。
body	String	消息体。
body_crc	Number	消息体校验和。
store_size	Number	存储大小。
property_list	Array of **property_list** objects	消息属性列表。
born_host	String	产生消息的主机IP。
store_host	String	存储消息的主机IP。
queue_id	Integer	队列ID。
queue_offset	Integer	在队列中的偏移量。

表 5-81 property_list

参数	参数类型	描述
name	String	属性名称。
value	String	属性值。

请求示例

查询消息。

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/messages?topic={topic}
```

响应示例

状态码： 200

查询消息成功。

```
{
  "messages" : [ {
    "msg_id" : "C0A807C900002774000000000000030",
    "instance_id" : "11c45539-xxxx-xxxx-xxxx-812c41f61f30",
    "topic" : "topic-test",
    "store_timestamp" : 1648888166319,
    "born_timestamp" : 1648888166275,
    "reconsume_times" : 0,
    "body" : "xxxx",
    "body_crc" : 1932557065,
    "store_size" : 175,
```

```
  "property_list" : [ {
    "name" : "KEYS",
    "value" : ""
  }, {
    "name" : "UNIQ_KEY",
    "value" : "7F00000123DC6E0BE85808B037820000"
  }, {
    "name" : "CLUSTER",
    "value" : "DmsCluster"
  }, {
    "name" : "TAGS",
    "value" : ""
  } ],
  "born_host" : "192.168.0.66:50098",
  "store_host" : "192.168.7.201:10100",
  "queue_id" : 2,
  "queue_offset" : 0
 } ],
 "total" : 1
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListMessagesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListMessagesRequest request = new ListMessagesRequest();
        request.withEngine(ListMessagesRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        try {
            ListMessagesResponse response = client.listMessages(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
```

```
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListMessagesRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        response = client.list_messages(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
```

```
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListMessagesRequest{}
    request.Engine = model.GetListMessagesRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    response, err := client.ListMessages(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询消息成功。

错误码

请参见**错误码**。

5.4.2 查询消息轨迹

功能介绍

查询消息轨迹。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/{project_id}/instances/{instance_id}/trace

表 5-82 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-83 Query 参数

参数	是否必选	参数类型	描述
msg_id	是	String	消息ID。

请求参数

无

响应参数

状态码： 200

表 5-84 响应 Body 参数

参数	参数类型	描述
trace	Array of **trace** objects	消息轨迹列表。

表 5-85 trace

参数	参数类型	描述
success	Boolean	是否成功。
trace_type	String	轨迹类型
timestamp	Number	时间。
group_name	String	生产组或消费组。
cost_time	Number	耗时。
request_id	String	请求ID。

参数	参数类型	描述
consume_status	Number	消费状态。 ● 0-消费成功 ● 1-消费超时 ● 2-消费发生异常 ● 3-消费返回NULL ● 5-消费失败
topic	String	主题名称。
msg_id	String	消息ID。
offset_msg_id	String	offset消息ID。
tags	String	消息的标签。
keys	String	消息的keys。
store_host	String	存储消息的主机IP。
client_host	String	产生消息的主机IP。
retry_times	Integer	重试次数。
body_length	Number	消息体长度。
msg_type	String	消息类型。
transaction_state	String	事务状态。
transaction_id	String	事务ID。
from_transaction_check	Boolean	是否为事务回查的响应。

请求示例

查询RocketMQ实例的消息轨迹。

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/trace?msg_id={msg_id}
```

响应示例

状态码： 200

查询成功。

```
[ {
  "success" : true,
  "trace_type" : "Pub",
  "timestamp" : 1634822858013,
  "group_name" : "ProducerGroupName",
  "cost_time" : 47,
  "request_id" : "644F0069C829287CBBF26B9A54390000",
```

```
  "consume_status" : 0,
  "topic" : "aaaaa",
  "msg_id" : "7F000001561018B4AAC26B9A0D1D0004",
  "offset_msg_id" : "C0A801170000277400000000000BE12",
  "tags" : "TagA",
  "keys" : "OrderID188",
  "store_host" : "192.168.0.1:10101",
  "client_host" : "127.0.0.1",
  "retry_times" : 0,
  "body_length" : 11,
  "msg_type" : "Normal_Msg",
  "transaction_state" : null,
  "transaction_id" : null,
  "from_transaction_check" : false
} ]
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListMessageTraceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListMessageTraceRequest request = new ListMessageTraceRequest();
        request.withEngine(ListMessageTraceRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        try {
            ListMessageTraceResponse response = client.listMessageTrace(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
```

```
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListMessageTraceRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        response = client.list_message_trace(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
```

```
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListMessageTraceRequest{}
    request.Engine = model.GetListMessageTraceRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    response, err := client.ListMessageTrace(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询成功。

错误码

请参见**错误码**。

5.4.3 导出死信消息

功能介绍

导出死信消息。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/messages/export

表 5-86 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-87 请求 Body 参数

参数	是否必选	参数类型	描述
topic	否	String	主题名称。
msg_id_list	否	Array of strings	消息ID列表。
uniq_key_list	否	Array of strings	唯一Key列表。

响应参数

状态码： 200

表 5-88 响应 Body 参数

参数	参数类型	描述
[数组元素]	Array of **Message** objects	消息。

表 5-89 Message

参数	参数类型	描述
msg_id	String	消息ID。
instance_id	String	实例ID。
topic	String	主题名称。
store_timestamp	Number	存储消息的时间。
born_timestamp	Number	产生消息的时间。

参数	参数类型	描述
reconsume_times	Integer	重试次数。
body	String	消息体。
body_crc	Number	消息体校验和。
store_size	Number	存储大小。
property_list	Array of **property_list** objects	消息属性列表。
born_host	String	产生消息的主机IP。
store_host	String	存储消息的主机IP。
queue_id	Integer	队列ID。
queue_offset	Integer	在队列中的偏移量。

表 5-90 property_list

参数	参数类型	描述
name	String	属性名称。
value	String	属性值。

请求示例

导出主题中指定消息ID和Key的死信消息。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/messages/export

{
  "topic" : "%DLQ%group1",
  "msg_id_list" : [ "C0A801170000277400000000013B19D", "C0A801170000277400000000013B30F" ],
  "uniq_key_list" : [ "7F000001001C18B4AAC26B8AED170010", "7F000001001C18B4AAC26B8AEE030015" ]
}
```

响应示例

状态码： 200

导出死信消息成功。

```
{
  "topic" : "topic_01",
  "body" : "Hello world",
  "property_list" : {
    "name" : "ORIGIN_MESSAGE_ID",
    "value" : "C0A8005B00002775000000000000EBAE"
  },
  "msg_id" : "C0A8005B000027750000000000133A2",
```

```
  "instance_id" : "56055acb-3c3b-4481-aeab-10464086c2b4",
  "store_timestamp" : 1662690563020,
  "born_timestamp" : 1662690324415,
  "reconsume_times" : 5,
  "body_crc" : 198614610,
  "store_size" : 317,
  "born_host" : "10.58.233.224:63529",
  "store_host" : "192.168.0.91:10101",
  "queue_id" : 0,
  "queue_offset" : 0
}
```

SDK 代码示例

SDK代码示例如下。

Java

导出主题中指定消息ID和Key的死信消息。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class ExportDlqMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ExportDlqMessageRequest request = new ExportDlqMessageRequest();
        request.withInstanceId("{instance_id}");
        ExportDlqMessageReq body = new ExportDlqMessageReq();
        List<String> listbodyUniqKeyList = new ArrayList<>();
        listbodyUniqKeyList.add("7F000001001C18B4AAC26B8AED170010");
        listbodyUniqKeyList.add("7F000001001C18B4AAC26B8AEE030015");
        List<String> listbodyMsgIdList = new ArrayList<>();
        listbodyMsgIdList.add("C0A801170000277400000000013B19D");
        listbodyMsgIdList.add("C0A801170000277400000000013B30F");
        body.withUniqKeyList(listbodyUniqKeyList);
        body.withMsgIdList(listbodyMsgIdList);
        body.withTopic("%DLQ%group1");
```

```
        request.withBody(body);
        try {
            ExportDlqMessageResponse response = client.exportDlqMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

导出主题中指定消息ID和Key的死信消息。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ExportDlqMessageRequest()
        request.instance_id = "{instance_id}"
        listUniqKeyListbody = [
            "7F000001001C18B4AAC26B8AED170010",
            "7F000001001C18B4AAC26B8AEE030015"
        ]
        listMsgIdListbody = [
            "C0A801170000277400000000000013B19D",
            "C0A801170000277400000000000013B30F"
        ]
        request.body = ExportDlqMessageReq(
            uniq_key_list=listUniqKeyListbody,
            msg_id_list=listMsgIdListbody,
            topic="%DLQ%group1"
        )
        response = client.export_dlq_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

导出主题中指定消息ID和Key的死信消息。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ExportDlqMessageRequest{}
    request.InstanceId = "{instance_id}"
    var listUniqKeyListbody = []string{
        "7F000001001C18B4AAC26B8AED170010",
        "7F000001001C18B4AAC26B8AEE030015",
    }
    var listMsgIdListbody = []string{
        "C0A801170000277400000000013B19D",
        "C0A801170000277400000000013B30F",
    }
    topicExportDlqMessageReq:= "%DLQ%group1"
    request.Body = &model.ExportDlqMessageReq{
        UniqKeyList: &listUniqKeyListbody,
        MsgIdList: &listMsgIdListbody,
        Topic: &topicExportDlqMessageReq,
    }
    response, err := client.ExportDlqMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	导出死信消息成功。

错误码

请参见**错误码**。

5.4.4 重发死信消息

功能介绍

重发死信消息。

调用方法

请参见**如何调用API**。

URI

POST /v2/{engine}/{project_id}/instances/{instance_id}/messages/deadletter-resend

表 5-91 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-92 请求 Body 参数

参数	是否必选	参数类型	描述
topic	否	String	topic。
msg_id_list	否	Array of strings	消息列表。

响应参数

状态码： 200

表 5-93 响应 Body 参数

参数	参数类型	描述
resend_results	Array of **resend_results** objects	重发死信消息结果。

表 5-94 resend_results

参数	参数类型	描述
msg_id	String	消息ID。
error_code	String	错误码。
error_message	String	错误信息。

请求示例

重发死信消息。

```
POST https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/messages/deadletter-resend

{
  "topic" : "%DLQ%group1",
  "msg_id_list" : [ "id1" ]
}
```

响应示例

状态码： 200

重发死信消息成功。

```
{
  "resend_results" : [ {
    "msg_id" : "C0A8149E000027760000000000B6835",
    "error_code" : "DMS.00000000",
    "error_message" : "Success."
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

重发死信消息。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class SendDlqMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        SendDlqMessageRequest request = new SendDlqMessageRequest();
        request.withEngine(SendDlqMessageRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        DeadletterResendReq body = new DeadletterResendReq();
        List<String> listbodyMsgIdList = new ArrayList<>();
        listbodyMsgIdList.add("id1");
        body.withMsgIdList(listbodyMsgIdList);
        body.withTopic("%DLQ%group1");
        request.withBody(body);
        try {
            SendDlqMessageResponse response = client.sendDlqMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

重发死信消息。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
```

```
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = SendDlqMessageRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        listMsgIdListbody = [
            "id1"
        ]
        request.body = DeadletterResendReq(
            msg_id_list=listMsgIdListbody,
            topic="%DLQ%group1"
        )
        response = client.send_dlq_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

重发死信消息。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()
```

```
    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.SendDlqMessageRequest{}
    request.Engine = model.GetSendDlqMessageRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    var listMsgIdListbody = []string{
        "id1",
    }
    topicDeadletterResendReq:= "%DLQ%group1"
    request.Body = &model.DeadletterResendReq{
        MsgIdList: &listMsgIdListbody,
        Topic: &topicDeadletterResendReq,
    }
    response, err := client.SendDlqMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	重发死信消息成功。

错误码

请参见**错误码**。

5.4.5 消费验证

功能介绍

消费验证。

调用方法

请参见**如何调用API**。

URI

POST /v2/{engine}/{project_id}/instances/{instance_id}/messages/resend

表 5-95 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-96 请求 Body 参数

参数	是否必选	参数类型	描述
group	否	String	Group ID。
topic	否	String	消息所属topic。
client_id	否	String	客户端ID。
msg_id_list	否	Array of strings	消息列表。

响应参数

状态码： 200

表 5-97 响应 Body 参数

参数	参数类型	描述
resend_results	Array of **resend_results** objects	消费验证结果。

表 5-98 resend_results

参数	参数类型	描述
msg_id	String	消息ID。
error_code	String	错误码。
error_message	String	错误信息。

请求示例

消费验证。

```
POST https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/messages/resend

{
  "group" : "GID_test",
  "client_id" : "192.168.0.1",
  "msg_id_list" : [ "id1" ]
}
```

响应示例

状态码： 200

消费验证成功。

```
{
  "resend_results" : [ {
    "msg_id" : "C0A8149E0000277600000000000B6835",
    "error_code" : "DMS.00000000",
    "error_message" : "Success."
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

消费验证。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class ValidateConsumedMessageSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
```

```
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
        ValidateConsumedMessageRequest request = new ValidateConsumedMessageRequest();
        request.withEngine(ValidateConsumedMessageRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        ResendReq body = new ResendReq();
        List<String> listbodyMsgIdList = new ArrayList<>();
        listbodyMsgIdList.add("id1");
        body.withMsgIdList(listbodyMsgIdList);
        body.withClientId("192.168.0.1");
        body.withGroup("GID_test");
        request.withBody(body);
        try {
            ValidateConsumedMessageResponse response = client.validateConsumedMessage(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

消费验证。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ValidateConsumedMessageRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        listMsgIdListbody = [
            "id1"
        ]
        request.body = ResendReq(
            msg_id_list=listMsgIdListbody,
            client_id="192.168.0.1",
            group="GID_test"
```

```
        )
        response = client.validate_consumed_message(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

消费验证。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ValidateConsumedMessageRequest{}
    request.Engine = model.GetValidateConsumedMessageRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    var listMsgIdListbody = []string{
        "id1",
    }
    clientIdResendReq:= "192.168.0.1"
    groupResendReq:= "GID_test"
    request.Body = &model.ResendReq{
        MsgIdList: &listMsgIdListbody,
        ClientId: &clientIdResendReq,
        Group: &groupResendReq,
    }
    response, err := client.ValidateConsumedMessage(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	消费验证成功。

错误码

请参见**错误码**。

5.5 用户管理

5.5.1 创建用户

功能介绍

创建用户。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/users

表 5-99 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-100 请求 Body 参数

参数	是否必选	参数类型	描述
access_key	否	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	否	String	密钥。 8-32个字符。 至少包含以下字符中的3种： • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	否	String	IP白名单。
admin	否	Boolean	是否为管理员。
default_topic_perm	否	String	默认的主题权限。
default_group_perm	否	String	默认的消费组权限。
topic_perms	否	Array of **topic_perms** objects	特殊的主题权限。
group_perms	否	Array of **group_perms** objects	特殊的消费组权限。

表 5-101 topic_perms

参数	是否必选	参数类型	描述
name	否	String	主题名称。
perm	否	String	权限。

表 5-102 group_perms

参数	是否必选	参数类型	描述
name	否	String	消费组名称。
perm	否	String	权限。

响应参数

状态码： 200

表 5-103 响应 Body 参数

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。
default_topic_perm	String	默认的主题权限。
default_group_perm	String	默认的消费组权限。
topic_perms	Array of **topic_perms** objects	特殊的主题权限。
group_perms	Array of **group_perms** objects	特殊的消费组权限。

表 5-104 topic_perms

参数	参数类型	描述
name	String	主题名称。

参数	参数类型	描述
perm	String	权限。

表 5-105 group_perms

参数	参数类型	描述
name	String	消费组名称。
perm	String	权限。

请求示例

创建一个非管理员的用户，授予发布、订阅topic1和group1的权限。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/users

{
  "access_key" : "user_name",
  "secret_key" : "**************",
  "white_remote_address" : "",
  "admin" : false,
  "default_topic_perm" : "DENY",
  "default_group_perm" : "DENY",
  "topic_perms" : [ {
    "name" : "topic1",
    "perm" : "PUB|SUB"
  } ],
  "group_perms" : [ {
    "name" : "group1",
    "perm" : "PUB|SUB"
  } ]
}
```

响应示例

状态码： 200

创建成功。

```
{
  "access_key" : "test_01",
  "admin" : false,
  "default_group_perm" : "DENY",
  "default_topic_perm" : "SUB",
  "group_perms" : [ ],
  "secret_key" : "**************",
  "topic_perms" : [ ],
  "white_remote_address" : ""
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个非管理员的用户，授予发布、订阅topic1和group1的权限。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateUserSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateUserRequest request = new CreateUserRequest();
        request.withInstanceId("{instance_id}");
        User body = new User();
        List<UserGroupPerms> listbodyGroupPerms = new ArrayList<>();
        listbodyGroupPerms.add(
            new UserGroupPerms()
                .withName("group1")
                .withPerm(UserGroupPerms.PermEnum.fromValue("PUB|SUB"))
        );
        List<UserTopicPerms> listbodyTopicPerms = new ArrayList<>();
        listbodyTopicPerms.add(
            new UserTopicPerms()
                .withName("topic1")
                .withPerm(UserTopicPerms.PermEnum.fromValue("PUB|SUB"))
        );
        body.withGroupPerms(listbodyGroupPerms);
        body.withTopicPerms(listbodyTopicPerms);
        body.withDefaultGroupPerm(User.DefaultGroupPermEnum.fromValue("DENY"));
        body.withDefaultTopicPerm(User.DefaultTopicPermEnum.fromValue("DENY"));
        body.withAdmin(false);
        body.withWhiteRemoteAddress("");
        body.withSecretKey("**************");
        body.withAccessKey("user_name");
        request.withBody(body);
        try {
            CreateUserResponse response = client.createUser(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
```

```
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建一个非管理员的用户，授予发布、订阅topic1和group1的权限。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateUserRequest()
        request.instance_id = "{instance_id}"
        listGroupPermsbody = [
            UserGroupPerms(
                name="group1",
                perm="PUB|SUB"
            )
        ]
        listTopicPermsbody = [
            UserTopicPerms(
                name="topic1",
                perm="PUB|SUB"
            )
        ]
        request.body = User(
            group_perms=listGroupPermsbody,
            topic_perms=listTopicPermsbody,
            default_group_perm="DENY",
            default_topic_perm="DENY",
            admin=False,
            white_remote_address="",
            secret_key="**********",
            access_key="user_name"
        )
        response = client.create_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
```

```
        print(e.error_code)
        print(e.error_msg)
```

Go

创建一个非管理员的用户，授予发布、订阅topic1和group1的权限。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateUserRequest{}
    request.InstanceId = "{instance_id}"
    nameGroupPerms:= "group1"
    permGroupPerms:= model.GetUserGroupPermsPermEnum().PUB|SUB
    var listGroupPermsbody = []model.UserGroupPerms{
        {
            Name: &nameGroupPerms,
            Perm: &permGroupPerms,
        },
    }
    nameTopicPerms:= "topic1"
    permTopicPerms:= model.GetUserTopicPermsPermEnum().PUB|SUB
    var listTopicPermsbody = []model.UserTopicPerms{
        {
            Name: &nameTopicPerms,
            Perm: &permTopicPerms,
        },
    }
    defaultGroupPermUser:= model.GetUserDefaultGroupPermEnum().DENY
    defaultTopicPermUser:= model.GetUserDefaultTopicPermEnum().DENY
    adminUser:= false
    whiteRemoteAddressUser:= ""
    secretKeyUser:= "**********"
    accessKeyUser:= "user_name"
    request.Body = &model.User{
        GroupPerms: &listGroupPermsbody,
        TopicPerms: &listTopicPermsbody,
        DefaultGroupPerm: &defaultGroupPermUser,
        DefaultTopicPerm: &defaultTopicPermUser,
        Admin: &adminUser,
```

```
        WhiteRemoteAddress: &whiteRemoteAddressUser,
        SecretKey: &secretKeyUser,
        AccessKey: &accessKeyUser,
    }
    response, err := client.CreateUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	创建成功。

错误码

请参见**错误码**。

5.5.2 查询用户列表

功能介绍

查询用户列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/users

表 5-106 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-107 Query 参数

参数	是否必选	参数类型	描述
offset	否	String	偏移量，表示从此偏移量开始查询， offset大于等于0。
limit	否	String	查询数量。

请求参数

无

响应参数

状态码： 200

表 5-108 响应 Body 参数

参数	参数类型	描述
users	Array of **User** objects	用户列表。
total	Number	总用户个数。

表 5-109 User

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： ● 大写字母 ● 小写字母 ● 数字 ● 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。
default_topic_perm	String	默认的主题权限。

参数	参数类型	描述
default_group_perm	String	默认的消费组权限。
topic_perms	Array of **topic_perms** objects	特殊的主题权限。
group_perms	Array of **group_perms** objects	特殊的消费组权限。

表 5-110 topic_perms

参数	参数类型	描述
name	String	主题名称。
perm	String	权限。

表 5-111 group_perms

参数	参数类型	描述
name	String	消费组名称。
perm	String	权限。

请求示例

查询用户列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/users?offset=0&limit=10
```

响应示例

状态码： 200

查询结果。

```
{
  "users" : [ {
    "access_key" : "test_01",
    "admin" : false,
    "default_group_perm" : "DENY",
    "default_topic_perm" : "SUB",
    "group_perms" : [ ],
    "secret_key" : "**********",
    "topic_perms" : [ ],
    "white_remote_address" : ""
  } ],
  "total" : 1
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListUserSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListUserRequest request = new ListUserRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListUserResponse response = client.listUser(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
```

```
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListUserRequest()
        request.instance_id = "{instance_id}"
        response = client.list_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListUserRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```

```
  }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询结果。

错误码

请参见**错误码**。

5.5.3 修改用户参数

功能介绍

修改用户参数。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/users/{user_name}

表 5-112 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
user_name	是	String	用户名。

请求参数

表 5-113 请求 Body 参数

参数	是否必选	参数类型	描述
access_key	否	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	否	String	密钥。 8-32个字符。 至少包含以下字符中的3种： • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	否	String	IP白名单。
admin	否	Boolean	是否为管理员。
default_topic_perm	否	String	默认的主题权限。
default_group_perm	否	String	默认的消费组权限。
topic_perms	否	Array of **topic_perms** objects	特殊的主题权限。
group_perms	否	Array of **group_perms** objects	特殊的消费组权限。

表 5-114 topic_perms

参数	是否必选	参数类型	描述
name	否	String	主题名称。
perm	否	String	权限。

表 5-115 group_perms

参数	是否必选	参数类型	描述
name	否	String	消费组名称。
perm	否	String	权限。

响应参数

状态码： 200

表 5-116 响应 Body 参数

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。
default_topic_perm	String	默认的主题权限。
default_group_perm	String	默认的消费组权限。
topic_perms	Array of **topic_perms** objects	特殊的主题权限。
group_perms	Array of **group_perms** objects	特殊的消费组权限。

表 5-117 topic_perms

参数	参数类型	描述
name	String	主题名称。

参数	参数类型	描述
perm	String	权限。

表 5-118 group_perms

参数	参数类型	描述
name	String	消费组名称。
perm	String	权限。

请求示例

修改用户参数，授予user_name用户发布、订阅topic1和group1的权限。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/users/{user_name}

{
  "access_key" : "user_name",
  "secret_key" : "**************",
  "white_remote_address" : "",
  "admin" : false,
  "default_topic_perm" : "DENY",
  "default_group_perm" : "DENY",
  "topic_perms" : [ {
    "name" : "topic1",
    "perm" : "PUB|SUB"
  } ],
  "group_perms" : [ {
    "name" : "group1",
    "perm" : "PUB|SUB"
  } ]
}
```

响应示例

状态码： 200

修改成功。

```
{
  "access_key" : "test_01",
  "admin" : false,
  "default_group_perm" : "DENY",
  "default_topic_perm" : "SUB",
  "group_perms" : [ ],
  "secret_key" : "**************",
  "topic_perms" : [ ],
  "white_remote_address" : ""
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改用户参数，授予user_name用户发布、订阅topic1和group1的权限。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateUserSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateUserRequest request = new UpdateUserRequest();
        request.withInstanceId("{instance_id}");
        request.withUserName("{user_name}");
        User body = new User();
        List<UserGroupPerms> listbodyGroupPerms = new ArrayList<>();
        listbodyGroupPerms.add(
            new UserGroupPerms()
                .withName("group1")
                .withPerm(UserGroupPerms.PermEnum.fromValue("PUB|SUB"))
        );
        List<UserTopicPerms> listbodyTopicPerms = new ArrayList<>();
        listbodyTopicPerms.add(
            new UserTopicPerms()
                .withName("topic1")
                .withPerm(UserTopicPerms.PermEnum.fromValue("PUB|SUB"))
        );
        body.withGroupPerms(listbodyGroupPerms);
        body.withTopicPerms(listbodyTopicPerms);
        body.withDefaultGroupPerm(User.DefaultGroupPermEnum.fromValue("DENY"));
        body.withDefaultTopicPerm(User.DefaultTopicPermEnum.fromValue("DENY"));
        body.withAdmin(false);
        body.withWhiteRemoteAddress("");
        body.withSecretKey("**********");
        body.withAccessKey("user_name");
        request.withBody(body);
        try {
            UpdateUserResponse response = client.updateUser(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
```

```
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改用户参数，授予user_name用户发布、订阅topic1和group1的权限。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateUserRequest()
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        listGroupPermsbody = [
            UserGroupPerms(
                name="group1",
                perm="PUB|SUB"
            )
        ]
        listTopicPermsbody = [
            UserTopicPerms(
                name="topic1",
                perm="PUB|SUB"
            )
        ]
        request.body = User(
            group_perms=listGroupPermsbody,
            topic_perms=listTopicPermsbody,
            default_group_perm="DENY",
            default_topic_perm="DENY",
            admin=False,
            white_remote_address="",
            secret_key="**********",
            access_key="user_name"
        )
        response = client.update_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
```

```
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改用户参数，授予user_name用户发布、订阅topic1和group1的权限。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateUserRequest{}
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    nameGroupPerms:= "group1"
    permGroupPerms:= model.GetUserGroupPermsPermEnum().PUB|SUB
    var listGroupPermsbody = []model.UserGroupPerms{
        {
            Name: &nameGroupPerms,
            Perm: &permGroupPerms,
        },
    }
    nameTopicPerms:= "topic1"
    permTopicPerms:= model.GetUserTopicPermsPermEnum().PUB|SUB
    var listTopicPermsbody = []model.UserTopicPerms{
        {
            Name: &nameTopicPerms,
            Perm: &permTopicPerms,
        },
    }
    defaultGroupPermUser:= model.GetUserDefaultGroupPermEnum().DENY
    defaultTopicPermUser:= model.GetUserDefaultTopicPermEnum().DENY
    adminUser:= false
    whiteRemoteAddressUser:= ""
    secretKeyUser:= "**********"
    accessKeyUser:= "user_name"
    request.Body = &model.User{
        GroupPerms: &listGroupPermsbody,
        TopicPerms: &listTopicPermsbody,
```

```
        DefaultGroupPerm: &defaultGroupPermUser,
        DefaultTopicPerm: &defaultTopicPermUser,
        Admin: &adminUser,
        WhiteRemoteAddress: &whiteRemoteAddressUser,
        SecretKey: &secretKeyUser,
        AccessKey: &accessKeyUser,
    }
    response, err := client.UpdateUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	修改成功。

错误码

请参见**错误码**。

5.5.4 查询用户详情

功能介绍

查询用户详情。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/users/{user_name}

表 5-119 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
user_name	是	String	用户名。

请求参数

无

响应参数

状态码： 200

表 5-120 响应 Body 参数

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#$%^&*()-_=+|[{}];:'",<.>/?密钥。 不能与名称或倒序的名称相同。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。
default_topic_perm	String	默认的主题权限。
default_group_perm	String	默认的消费组权限。
topic_perms	Array of **topic_perms** objects	特殊的主题权限。
group_perms	Array of **group_perms** objects	特殊的消费组权限。

表 5-121 topic_perms

参数	参数类型	描述
name	String	主题名称。
perm	String	权限。

表 5-122 group_perms

参数	参数类型	描述
name	String	消费组名称。
perm	String	权限。

请求示例

查询用户详情。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/users/{user_name}?offset=0&limit=10
```

响应示例

状态码： 200

查询用户详情成功。

```
{
  "access_key" : "test_01",
  "admin" : false,
  "default_group_perm" : "DENY",
  "default_topic_perm" : "SUB",
  "group_perms" : [ ],
  "secret_key" : "**************",
  "topic_perms" : [ ],
  "white_remote_address" : ""
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowUserSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
```

```
            .withProjectId(projectId)
            .withAk(ak)
            .withSk(sk);

    rocketmqClient client = rocketmqClient.newBuilder()
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
    ShowUserRequest request = new ShowUserRequest();
    request.withInstanceId("{instance_id}");
    request.withUserName("{user_name}");
    try {
        ShowUserResponse response = client.showUser(request);
        System.out.println(response.toString());
    } catch (ConnectionException e) {
        e.printStackTrace();
    } catch (RequestTimeoutException e) {
        e.printStackTrace();
    } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(e.getHttpStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
  }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowUserRequest()
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        response = client.show_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowUserRequest{}
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    response, err := client.ShowUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询用户详情成功。

错误码

请参见**错误码**。

5.5.5 删除用户

功能介绍

删除用户。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}/users/{user_name}

表 5-123 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
user_name	是	String	用户名。

请求参数

无

响应参数

状态码： 200

表 5-124 响应 Body 参数

参数	参数类型	描述
message	String	信息。

请求示例

删除指定的用户。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/users/{user_name}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class DeleteUserSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteUserRequest request = new DeleteUserRequest();
        request.withInstanceId("{instance_id}");
        request.withUserName("{user_name}");
        try {
            DeleteUserResponse response = client.deleteUser(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteUserRequest()
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        response = client.delete_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteUserRequest{}
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    response, err := client.DeleteUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
```

```
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	删除成功。

错误码

请参见**错误码**。

5.5.6 查询主题的授权用户列表

功能介绍

查询主题的授权用户列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/topics/{topic}/accesspolicy

表 5-125 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
topic	是	String	主题名称。

表 5-126 Query 参数

参数	是否必选	参数类型	描述
offset	否	String	偏移量，表示从此偏移量开始查询， offset大于等于0。
limit	否	String	查询数量。

请求参数

无

响应参数

状态码： 200

表 5-127 响应 Body 参数

参数	参数类型	描述
policies	Array of **policies** objects	用户列表。
total	Number	总用户个数。
name	String	主题或消费组名称。

表 5-128 policies

参数	参数类型	描述
access_key	String	用户名。
secret_key	String	密钥。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。
perm	String	权限。

请求示例

查询主题的授权用户列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/topics/{topic}/accesspolicy
```

响应示例

状态码： 200

查询主题的授权用户列表成功

```
{
  "policies" : [ ],
  "total" : 0,
  "name" : "test"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListTopicAccessPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListTopicAccessPolicyRequest request = new ListTopicAccessPolicyRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ListTopicAccessPolicyResponse response = client.listTopicAccessPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
```

```
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListTopicAccessPolicyRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.list_topic_access_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()
```

```
    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListTopicAccessPolicyRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ListTopicAccessPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询主题的授权用户列表成功

错误码

请参见**错误码**。

5.5.7 查询消费组的授权用户列表

功能介绍

查询消费组的授权用户列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/{project_id}/instances/{instance_id}/groups/{group_id}/accesspolicy

表 5-129 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。 缺省值：**reliability**

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。
group_id	是	String	消费组。

表 5-130 Query 参数

参数	是否必选	参数类型	描述
offset	否	String	偏移量，表示从此偏移量开始查询， offset大于等于0。
limit	否	String	查询数量。

请求参数

无

响应参数

状态码： 200

表 5-131 响应 Body 参数

参数	参数类型	描述
policies	Array of **policies** objects	用户列表。
total	Number	总用户个数。
name	String	主题或消费组名称。

表 5-132 policies

参数	参数类型	描述
access_key	String	用户名。
secret_key	String	密钥。
white_remote_address	String	IP白名单。
admin	Boolean	是否为管理员。

参数	参数类型	描述
perm	String	权限。

请求示例

查询消费组的授权用户列表。

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/groups/{group_id}/accesspolicy
```

响应示例

状态码： 200

消费组的授权用户列表成功。

```
{
  "policies" : {
    "access_key" : "test_01",
    "secret_key" : "**********",
    "white_remote_address" : "",
    "admin" : false,
    "perm" : ""
  },
  "total" : 1,
  "name" : "test"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListConsumeGroupAccessPolicySolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);
```

```
        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListConsumeGroupAccessPolicyRequest request = new ListConsumeGroupAccessPolicyRequest();
        request.withEngine(ListConsumeGroupAccessPolicyRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        request.withGroupId("{group_id}");
        try {
            ListConsumeGroupAccessPolicyResponse response = client.listConsumeGroupAccessPolicy(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListConsumeGroupAccessPolicyRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.group_id = "{group_id}"
        response = client.list_consume_group_access_policy(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListConsumeGroupAccessPolicyRequest{}
    request.Engine = model.GetListConsumeGroupAccessPolicyRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    request.GroupId = "{group_id}"
    response, err := client.ListConsumeGroupAccessPolicy(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	消费组的授权用户列表成功。

错误码

请参见**错误码**。

5.6 元数据迁移

5.6.1 新建元数据迁移任务

功能介绍

新建元数据迁移任务。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/metadata

表 5-133 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-134 Query 参数

参数	是否必选	参数类型	描述
overwrite	是	String	true开启同名覆盖，会对已有的同名元数据的配置进行修改，false时当topic或group已存在则会报错。
name	是	String	迁移任务名称，名称规则参考创建实例
type	是	String	迁移任务类型，分为自建RocketMQ上云(rocketmq)、自建RabbitMQ上云(rabbitToRocket)

请求参数

表 5-135 请求 Body 参数

参数	是否必选	参数类型	描述
topicConfigTable	否	Map<String,**MigrationRocketMqTopicConfig**>	RocketMQ topic 元数据，键为topic名，值为topic配置，迁移任务类型为自建RocketMQ上云(rocketmq)时必填。
subscriptionGroupTable	否	Map<String,**MigrationRocketMqSubscriptionGroup**>	RocketMQ消费组元数据，键为消费组名，值为消费组配置，迁移任务类型为自建RocketMQ上云(rocketmq)时必填。
vhosts	否	Array of **MigrationRabbitVhostMetadata** objects	RabbitMQ vhost元数据列表，迁移任务类型为自建RabbitMQ上云(rabbitToRocket)时必填。
queues	否	Array of **MigrationRabbitQueueMetadata** objects	RabbitMQ队列元数据列表，迁移任务类型为自建RabbitMQ上云(rabbitToRocket)时必填。
exchanges	否	Array of **MigrationRabbitExchangeMetadata** objects	RabbitMQ交换机元数据列表，迁移任务类型为自建RabbitMQ上云(rabbitToRocket)时必填。
bindings	否	Array of **MigrationRabbitBindingMetadata** objects	RabbitMQ binding元数据列表，迁移任务类型为自建RabbitMQ上云(rabbitToRocket)时必填。

表 5-136 MigrationRocketMqTopicConfig

参数	是否必选	参数类型	描述
topicName	否	String	topic名称。
order	否	Boolean	是否有序消息。 缺省值：**false**
perm	否	Integer	topic权限。 缺省值：**6**

参数	是否必选	参数类型	描述
readQueueNums	否	Integer	读队列个数。 缺省值：**16**
writeQueueNums	否	Integer	写队列个数。 缺省值：**16**
topicFilterType	否	String	topic过滤类型。 ● SINGLE_TAG：单标签 ● MULTI_TAG：多标签 缺省值：**SINGLE_TAG**
topicSysFlag	否	Integer	topic系统标志位。 缺省值：**0**

表 5-137 MigrationRocketMqSubscriptionGroup

参数	是否必选	参数类型	描述
groupName	否	String	消费组名。
consumeBroadcastEnable	否	Boolean	是否允许以广播模式消费。 缺省值：**true**
consumeEnable	否	Boolean	是否允许消费。 缺省值：**true**
consumeFromMinEnable	否	Boolean	是否从最小偏移量开始消费。 缺省值：**true**
notifyConsumerIdsChangedEnable	否	Boolean	消费者ID变化时是否通知。 缺省值：**true**
retryMaxTimes	否	Integer	消费最大重试次数。 缺省值：**16**
retryQueueNums	否	Integer	重试队列个数。 缺省值：**1**
whichBrokerWhenConsumeSlow	否	Long	慢消费时选择的broker节点ID。 缺省值：**1**

表 5-138 MigrationRabbitVhostMetadata

参数	是否必选	参数类型	描述
name	否	String	vhost名称。

表 5-139 MigrationRabbitQueueMetadata

参数	是否必选	参数类型	描述
vhost	否	String	vhost名称。
name	否	String	队列名称。
durable	否	Boolean	是否持久化。

表 5-140 MigrationRabbitExchangeMetadata

参数	是否必选	参数类型	描述
vhost	否	String	vhost名称。
name	否	String	交换机名称。
type	否	String	交换机类型。
durable	否	Boolean	是否持久化。

表 5-141 MigrationRabbitBindingMetadata

参数	是否必选	参数类型	描述
vhost	否	String	vhost名称。
source	否	String	消息的来源。
destination	否	String	消息的目标。
destination_type	否	String	目标的类型。
routing_key	否	String	路由键。

响应参数

状态码： 200

表 5-142 响应 Body 参数

参数	参数类型	描述
task_id	String	任务ID

请求示例

- 创建元数据迁移任务，迁移其他厂商或自建RocketMQ实例的元数据到云上RocketMQ实例。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/metadata?
overwrite=true&name=task-123&type=rocketmq

{
  "topicConfigTable" : {
    "topic-test1" : {
      "order" : false,
      "perm" : 6,
      "readQueueNums" : 3,
      "topicFilterType" : "SINGLE_TAG",
      "topicName" : "topic-test1",
      "topicSysFlag" : 0,
      "writeQueueNums" : 3
    }
  },
  "subscriptionGroupTable" : {
    "group-test1" : {
      "consumeBroadcastEnable" : true,
      "consumeEnable" : true,
      "consumeFromMinEnable" : true,
      "groupName" : "group-test1",
      "notifyConsumerIdsChangedEnable" : true,
      "retryMaxTimes" : 2,
      "retryQueueNums" : 1,
      "whichBrokerWhenConsumeSlow" : 1
    }
  }
}
```

- 创建元数据迁移任务，迁移RabbitMQ的元数据到云上RocketMQ实例。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/metadata?
overwrite=true&name=task-123&type=rabbitToRocket

{
  "vhosts" : [ {
    "name" : "DeleteVhost123"
  } ],
  "queues" : [ {
    "name" : "test-001",
    "vhost" : "/",
    "durable" : false
  } ],
  "exchanges" : [ {
    "name" : "direct",
    "vhost" : "/",
    "type" : "topic",
    "durable" : false
  } ],
  "bindings" : [ {
    "source" : "direct",
    "vhost" : "/",
    "destination" : "test-001",
    "destination_type" : "queue",
    "routing_key" : "test-001"
```

```
 } ]
}
```

响应示例

状态码： 200

创建成功

```
{
  "task_id" : "6cf4dcd3-8471-4139-8b5b-8a3a71f704c7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建元数据迁移任务，迁移其他厂商或自建RocketMQ实例的元数据到云上RocketMQ实例。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.Map;
import java.util.HashMap;

public class CreateRocketMqMigrationTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateRocketMqMigrationTaskRequest request = new CreateRocketMqMigrationTaskRequest();
        request.withInstanceId("{instance_id}");
        CreateRocketMqMigrationTaskReq body = new CreateRocketMqMigrationTaskReq();
        MigrationRocketMqSubscriptionGroup subscriptionGroupTableSubscriptionGroupTable = new MigrationRocketMqSubscriptionGroup();
        subscriptionGroupTableSubscriptionGroupTable.withGroupName("group-test1")
            .withConsumeBroadcastEnable(true)
            .withConsumeEnable(true)
            .withConsumeFromMinEnable(true)
```

```
            .withNotifyConsumerIdsChangedEnable(true)
            .withRetryMaxTimes(2)
            .withRetryQueueNums(1)
            .withWhichBrokerWhenConsumeSlow(1L);
        Map<String, MigrationRocketMqSubscriptionGroup> listbodySubscriptionGroupTable = new
HashMap<>();
        listbodySubscriptionGroupTable.put("group-test1",
subscriptionGroupTableSubscriptionGroupTable);
        MigrationRocketMqTopicConfig topicConfigTableTopicConfigTable = new
MigrationRocketMqTopicConfig();
        topicConfigTableTopicConfigTable.withTopicName("topic-test1")
            .withOrder(false)
            .withPerm(6)
            .withReadQueueNums(3)
            .withWriteQueueNums(3)
            .withTopicFilterType("SINGLE_TAG")
            .withTopicSysFlag(0);
        Map<String, MigrationRocketMqTopicConfig> listbodyTopicConfigTable = new HashMap<>();
        listbodyTopicConfigTable.put("topic-test1", topicConfigTableTopicConfigTable);
        body.withSubscriptionGroupTable(listbodySubscriptionGroupTable);
        body.withTopicConfigTable(listbodyTopicConfigTable);
        request.withBody(body);
        try {
            CreateRocketMqMigrationTaskResponse response =
client.createRocketMqMigrationTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

- 创建元数据迁移任务，迁移RabbitMQ的元数据到云上RocketMQ实例。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class CreateRocketMqMigrationTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great
security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or
environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before
running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local
environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
```

```
            .withAk(ak)
            .withSk(sk);

    rocketmqClient client = rocketmqClient.newBuilder()
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
    CreateRocketMqMigrationTaskRequest request = new CreateRocketMqMigrationTaskRequest();
    request.withInstanceId("{instance_id}");
    CreateRocketMqMigrationTaskReq body = new CreateRocketMqMigrationTaskReq();
    List<MigrationRabbitBindingMetadata> listbodyBindings = new ArrayList<>();
    listbodyBindings.add(
        new MigrationRabbitBindingMetadata()
            .withVhost("/")
            .withSource("direct")
            .withDestination("test-001")
            .withDestinationType("queue")
            .withRoutingKey("test-001")
    );
    List<MigrationRabbitExchangeMetadata> listbodyExchanges = new ArrayList<>();
    listbodyExchanges.add(
        new MigrationRabbitExchangeMetadata()
            .withVhost("/")
            .withName("direct")
            .withType("topic")
            .withDurable(false)
    );
    List<MigrationRabbitQueueMetadata> listbodyQueues = new ArrayList<>();
    listbodyQueues.add(
        new MigrationRabbitQueueMetadata()
            .withVhost("/")
            .withName("test-001")
            .withDurable(false)
    );
    List<MigrationRabbitVhostMetadata> listbodyVhosts = new ArrayList<>();
    listbodyVhosts.add(
        new MigrationRabbitVhostMetadata()
            .withName("DeleteVhost123")
    );
    body.withBindings(listbodyBindings);
    body.withExchanges(listbodyExchanges);
    body.withQueues(listbodyQueues);
    body.withVhosts(listbodyVhosts);
    request.withBody(body);
    try {
        CreateRocketMqMigrationTaskResponse response =
client.createRocketMqMigrationTask(request);
        System.out.println(response.toString());
    } catch (ConnectionException e) {
        e.printStackTrace();
    } catch (RequestTimeoutException e) {
        e.printStackTrace();
    } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(e.getHttpStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
  }
}
```

Python

- 创建元数据迁移任务，迁移其他厂商或自建RocketMQ实例的元数据到云上RocketMQ实例。

```
# coding: utf-8
```

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateRocketMqMigrationTaskRequest()
        request.instance_id = "{instance_id}"
        subscriptionGroupTableSubscriptionGroupTable = MigrationRocketMqSubscriptionGroup(
            group_name="group-test1",
            consume_broadcast_enable=True,
            consume_enable=True,
            consume_from_min_enable=True,
            notify_consumer_ids_changed_enable=True,
            retry_max_times=2,
            retry_queue_nums=1,
            which_broker_when_consume_slow=1
        )
        listSubscriptionGroupTablebody = {
            "group-test1": subscriptionGroupTableSubscriptionGroupTable
        }
        topicConfigTableTopicConfigTable = MigrationRocketMqTopicConfig(
            topic_name="topic-test1",
            order=False,
            perm=6,
            read_queue_nums=3,
            write_queue_nums=3,
            topic_filter_type="SINGLE_TAG",
            topic_sys_flag=0
        )
        listTopicConfigTablebody = {
            "topic-test1": topicConfigTableTopicConfigTable
        }
        request.body = CreateRocketMqMigrationTaskReq(
            subscription_group_table=listSubscriptionGroupTablebody,
            topic_config_table=listTopicConfigTablebody
        )
        response = client.create_rocket_mq_migration_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 创建元数据迁移任务，迁移RabbitMQ的元数据到云上RocketMQ实例。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *
```

```
if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateRocketMqMigrationTaskRequest()
        request.instance_id = "{instance_id}"
        listBindingsbody = [
            MigrationRabbitBindingMetadata(
                vhost="/",
                source="direct",
                destination="test-001",
                destination_type="queue",
                routing_key="test-001"
            )
        ]
        listExchangesbody = [
            MigrationRabbitExchangeMetadata(
                vhost="/",
                name="direct",
                type="topic",
                durable=False
            )
        ]
        listQueuesbody = [
            MigrationRabbitQueueMetadata(
                vhost="/",
                name="test-001",
                durable=False
            )
        ]
        listVhostsbody = [
            MigrationRabbitVhostMetadata(
                name="DeleteVhost123"
            )
        ]
        request.body = CreateRocketMqMigrationTaskReq(
            bindings=listBindingsbody,
            exchanges=listExchangesbody,
            queues=listQueuesbody,
            vhosts=listVhostsbody
        )
        response = client.create_rocket_mq_migration_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

- 创建元数据迁移任务，迁移其他厂商或自建RocketMQ实例的元数据到云上RocketMQ实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateRocketMqMigrationTaskRequest{}
    request.InstanceId = "{instance_id}"
    groupNameSubscriptionGroupTable:= "group-test1"
    consumeBroadcastEnableSubscriptionGroupTable:= true
    consumeEnableSubscriptionGroupTable:= true
    consumeFromMinEnableSubscriptionGroupTable:= true
    notifyConsumerIdsChangedEnableSubscriptionGroupTable:= true
    retryMaxTimesSubscriptionGroupTable:= int32(2)
    retryQueueNumsSubscriptionGroupTable:= int32(1)
    whichBrokerWhenConsumeSlowSubscriptionGroupTable:= int64(1)
    subscriptionGroupTableSubscriptionGroupTable := model.MigrationRocketMqSubscriptionGroup{
        GroupName: &groupNameSubscriptionGroupTable,
        ConsumeBroadcastEnable: &consumeBroadcastEnableSubscriptionGroupTable,
        ConsumeEnable: &consumeEnableSubscriptionGroupTable,
        ConsumeFromMinEnable: &consumeFromMinEnableSubscriptionGroupTable,
        NotifyConsumerIdsChangedEnable: &notifyConsumerIdsChangedEnableSubscriptionGroupTable,
        RetryMaxTimes: &retryMaxTimesSubscriptionGroupTable,
        RetryQueueNums: &retryQueueNumsSubscriptionGroupTable,
        WhichBrokerWhenConsumeSlow: &whichBrokerWhenConsumeSlowSubscriptionGroupTable,
    }
    var listSubscriptionGroupTablebody = map[string](model.MigrationRocketMqSubscriptionGroup){
        "group-test1": subscriptionGroupTableSubscriptionGroupTable,
    }
    topicNameTopicConfigTable:= "topic-test1"
    orderTopicConfigTable:= false
    permTopicConfigTable:= int32(6)
    readQueueNumsTopicConfigTable:= int32(3)
    writeQueueNumsTopicConfigTable:= int32(3)
    topicFilterTypeTopicConfigTable:= "SINGLE_TAG"
    topicSysFlagTopicConfigTable:= int32(0)
    topicConfigTableTopicConfigTable := model.MigrationRocketMqTopicConfig{
        TopicName: &topicNameTopicConfigTable,
        Order: &orderTopicConfigTable,
        Perm: &permTopicConfigTable,
        ReadQueueNums: &readQueueNumsTopicConfigTable,
        WriteQueueNums: &writeQueueNumsTopicConfigTable,
```

```
        TopicFilterType: &topicFilterTypeTopicConfigTable,
        TopicSysFlag: &topicSysFlagTopicConfigTable,
    }
    var listTopicConfigTablebody = map[string](model.MigrationRocketMqTopicConfig){
        "topic-test1": topicConfigTableTopicConfigTable,
    }
    request.Body = &model.CreateRocketMqMigrationTaskReq{
        SubscriptionGroupTable: listSubscriptionGroupTablebody,
        TopicConfigTable: listTopicConfigTablebody,
    }
    response, err := client.CreateRocketMqMigrationTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建元数据迁移任务，迁移RabbitMQ的元数据到云上RocketMQ实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateRocketMqMigrationTaskRequest{}
    request.InstanceId = "{instance_id}"
    vhostBindings:= "/"
    sourceBindings:= "direct"
    destinationBindings:= "test-001"
    destinationTypeBindings:= "queue"
    routingKeyBindings:= "test-001"
    var listBindingsbody = []model.MigrationRabbitBindingMetadata{
        {
            Vhost: &vhostBindings,
            Source: &sourceBindings,
            Destination: &destinationBindings,
            DestinationType: &destinationTypeBindings,
            RoutingKey: &routingKeyBindings,
        },
    }
    vhostExchanges:= "/"
    nameExchanges:= "direct"
```

```
    typeExchanges:= "topic"
    durableExchanges:= false
    var listExchangesbody = []model.MigrationRabbitExchangeMetadata{
        {
            Vhost: &vhostExchanges,
            Name: &nameExchanges,
            Type: &typeExchanges,
            Durable: &durableExchanges,
        },
    }
    vhostQueues:= "/"
    nameQueues:= "test-001"
    durableQueues:= false
    var listQueuesbody = []model.MigrationRabbitQueueMetadata{
        {
            Vhost: &vhostQueues,
            Name: &nameQueues,
            Durable: &durableQueues,
        },
    }
    nameVhosts:= "DeleteVhost123"
    var listVhostsbody = []model.MigrationRabbitVhostMetadata{
        {
            Name: &nameVhosts,
        },
    }
    request.Body = &model.CreateRocketMqMigrationTaskReq{
        Bindings: &listBindingsbody,
        Exchanges: &listExchangesbody,
        Queues: &listQueuesbody,
        Vhosts: &listVhostsbody,
    }
    response, err := client.CreateRocketMqMigrationTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	创建成功

错误码

请参见**错误码**。

5.6.2 查询实例下所有迁移任务或查询指定迁移任务信息

功能介绍

1. 查询实例下所有迁移任务

2. 查询指定迁移任务信息

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/metadata

表 5-143 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-144 Query 参数

参数	是否必选	参数类型	描述
id	否	String	任务ID
type	否	String	查询类型
offset	否	String	当前页，从1开始
limit	否	String	当前页大小
name	否	String	• 查询vhost列表时，该字段可为空。 • 查询exchange列表时，该字段为exchange所属vhost名称。 • 查询queue列表时，该字段为queue所属vhost-所属exchange，例vhost1-exchange1。

请求参数

无

响应参数

状态码： 200

表 5-145 响应 Body 参数

参数	参数类型	描述
total	Integer	元数据迁移任务总数。
task	Array of MetadataTask objects	元数据迁移任务列表。

表 5-146 MetadataTask

参数	参数类型	描述
id	String	元数据迁移任务ID。
name	String	元数据迁移任务名称。
start_date	String	元数据迁移任务开始时间。
status	String	元数据迁移任务状态。
type	String	元数据迁移类型。

请求示例

查询RocketMQ实例下所有迁移任务。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/metadata
```

响应示例

状态码： 200

返回状态码

```
{
  "total" : 1,
  "task" : [ {
    "id" : "6cf4dcd3-8471-4139-8b5b-8a3a71f704c7",
    "name" : "task-932331847",
    "start_date" : "2023-03-13 19:43:32.12",
    "status" : "finished",
    "type" : "rabbitToRocket"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```

```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListRocketMqMigrationTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListRocketMqMigrationTaskRequest request = new ListRocketMqMigrationTaskRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListRocketMqMigrationTaskResponse response = client.listRocketMqMigrationTask(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \
```

```
    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListRocketMqMigrationTaskRequest()
        request.instance_id = "{instance_id}"
        response = client.list_rocket_mq_migration_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListRocketMqMigrationTaskRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListRocketMqMigrationTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	返回状态码

错误码

请参见**错误码**。

5.6.3 删除元数据迁移任务

功能介绍

删除元数据迁移任务。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}/metadata

表 5-147 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-148 请求 Body 参数

参数	是否必选	参数类型	描述
task_ids	是	Array of strings	需要删除的任务列表。

响应参数

状态码： 200

表 5-149 响应 Body 参数

参数	参数类型	描述
-	String	返回状态码。

请求示例

删除元数据迁移任务。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/metadata
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class DeleteRocketMqMigrationTaskSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteRocketMqMigrationTaskRequest request = new DeleteRocketMqMigrationTaskRequest();
        request.withInstanceId("{instance_id}");
        MetadataDeleteReq body = new MetadataDeleteReq();
        request.withBody(body);
        try {
            DeleteRocketMqMigrationTaskResponse response = client.deleteRocketMqMigrationTask(request);
            System.out.println(response.toString());
```

```
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteRocketMqMigrationTaskRequest()
        request.instance_id = "{instance_id}"
        request.body = MetadataDeleteReq(
        )
        response = client.delete_rocket_mq_migration_task(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
```

```
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteRocketMqMigrationTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.Body = &model.MetadataDeleteReq{
    }
    response, err := client.DeleteRocketMqMigrationTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	返回状态码

错误码

请参见**错误码**。

5.7 参数管理

5.7.1 查询 RocketMQ 配置

功能介绍

该接口用于查询RocketMQ配置，若成功则返回配置的相关信息。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/rocketmq/instances/{instance_id}/configs

表 5-150 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码： 200

表 5-151 响应 Body 参数

参数	参数类型	描述
rocketmq_configs	Array of **RocketMQConfigResp** objects	RocketMQ配置。

表 5-152 RocketMQConfigResp

参数	参数类型	描述
name	String	RocketMQ配置名称
value	String	RocketMQ配置当前值
config_type	String	RocketMQ配置的类型
default_value	String	RocketMQ配置的默认值
valid_values	String	RocketMQ配置取值的范围
value_type	String	RocketMQ配置值的类型

请求示例

```
GET https://{endpoint}/v2/{project_id}/rocketmq/instances/{instance_id}/configs
```

响应示例

状态码： 200

查询RocketMQ配置成功。

```
{
  "rocketmq_configs" : [ {
    "name" : "fileReservedTime",
    "value" : 48,
    "config_type" : "dynamic",
    "default_value" : 48,
    "valid_values" : "0-720",
    "value_type" : "integer"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowRocketMqConfigsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowRocketMqConfigsRequest request = new ShowRocketMqConfigsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowRocketMqConfigsResponse response = client.showRocketMqConfigs(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
```

```
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowRocketMqConfigsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_rocket_mq_configs(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
```

```
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowRocketMqConfigsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowRocketMqConfigs(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询RocketMQ配置成功。

错误码

请参见**错误码**。

5.7.2 修改 RocketMQ 配置

功能介绍

该接口用于修改RocketMQ配置。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/rocketmq/instances/{instance_id}/configs

表 5-153 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。

参数	是否必选	参数类型	描述
instance_id	是	String	实例ID。

请求参数

表 5-154 请求 Body 参数

参数	是否必选	参数类型	描述
rocketmq_configs	否	Array of **RocketMQConfigReq** objects	RocketMQ配置。

表 5-155 RocketMQConfigReq

参数	是否必选	参数类型	描述
name	否	String	RocketMQ配置名称
value	否	String	RocketMQ配置目标值

响应参数

无

请求示例

修改RocketMQ配置的参数，将文件保留时间修改为72小时。

```
PUT  https://{endpoint}/v2/{project_id}/rocketmq/instances/{instance_id}/configs

{
  "rocketmq_configs" : [ {
    "name" : "fileReservedTime",
    "value" : 72
  } ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改RocketMQ配置的参数，将文件保留时间修改为72小时。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class UpdateRocketMqConfigsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateRocketMqConfigsRequest request = new UpdateRocketMqConfigsRequest();
        request.withInstanceId("{instance_id}");
        ModifyConfigReq body = new ModifyConfigReq();
        List<RocketMQConfigReq> listbodyRocketmqConfigs = new ArrayList<>();
        listbodyRocketmqConfigs.add(
            new RocketMQConfigReq()
                .withName(RocketMQConfigReq.NameEnum.fromValue("fileReservedTime"))
                .withValue("72")
        );
        body.withRocketmqConfigs(listbodyRocketmqConfigs);
        request.withBody(body);
        try {
            UpdateRocketMqConfigsResponse response = client.updateRocketMqConfigs(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

修改RocketMQ配置的参数，将文件保留时间修改为72小时。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateRocketMqConfigsRequest()
        request.instance_id = "{instance_id}"
        listRocketmqConfigsbody = [
            RocketMQConfigReq(
                name="fileReservedTime",
                value="72"
            )
        ]
        request.body = ModifyConfigReq(
            rocketmq_configs=listRocketmqConfigsbody
        )
        response = client.update_rocket_mq_configs(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

修改RocketMQ配置的参数，将文件保留时间修改为72小时。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
```

```
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateRocketMqConfigsRequest{}
    request.InstanceId = "{instance_id}"
    nameRocketmqConfigs:= model.GetRocketMqConfigReqNameEnum().FILE_RESERVED_TIME
    valueRocketmqConfigs:= "72"
    var listRocketmqConfigsbody = []model.RocketMqConfigReq{
        {
            Name: &nameRocketmqConfigs,
            Value: &valueRocketmqConfigs,
        },
    }
    request.Body = &model.ModifyConfigReq{
        RocketmqConfigs: &listRocketmqConfigsbody,
    }
    response, err := client.UpdateRocketMqConfigs(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	修改RocketMQ配置成功。

错误码

请参见**错误码**。

5.8 标签管理

5.8.1 批量添加或删除实例标签

功能介绍

批量添加或删除实例标签。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/rocketmq/{instance_id}/tags/action

表 5-156 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-157 请求 Body 参数

参数	是否必选	参数类型	描述
action	否	String	操作标识（仅支持小写）： • create（创建） • delete（删除）
tags	否	Array of **TagEntity** objects	标签列表。

表 5-158 TagEntity

参数	是否必选	参数类型	描述
key	否	String	标签键。 ● 不能为空。 ● 对于同一个实例，Key值唯一。 ● 长度为1~128个字符（中文也可以输入128个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持 _ . : = + - @ ● 首尾字符不能为空格。
value	否	String	标签值。 ● 长度为0~255个字符（中文也可以输入255个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持 _ . : = + - @ ● 首尾字符不能为空格。

响应参数

无

请求示例

创建两个实例标签，标签名为key1、key2，值为value1、value2。

```
POST https://{endpoint}/v2/{project_id}/rocketmq/{instance_id}/tags/action

{
  "action" : "create",
  "tags" : [ {
    "key" : "key1",
    "value" : "value1"
  }, {
    "key" : "key2",
    "value" : "value2"
  } ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

创建两个实例标签，标签名为key1、key2，值为value1、value2。

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;

import java.util.List;
import java.util.ArrayList;

public class BatchCreateOrDeleteRocketmqTagSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchCreateOrDeleteRocketmqTagRequest request = new BatchCreateOrDeleteRocketmqTagRequest();
        request.withInstanceId("{instance_id}");
        BatchCreateOrDeleteTagReq body = new BatchCreateOrDeleteTagReq();
        List<TagEntity> listbodyTags = new ArrayList<>();
        listbodyTags.add(
            new TagEntity()
                .withKey("key1")
                .withValue("value1")
        );
        listbodyTags.add(
            new TagEntity()
                .withKey("key2")
                .withValue("value2")
        );
        body.withTags(listbodyTags);
        body.withAction(BatchCreateOrDeleteTagReq.ActionEnum.fromValue("create"));
        request.withBody(body);
        try {
            BatchCreateOrDeleteRocketmqTagResponse response = client.batchCreateOrDeleteRocketmqTag(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
```

```
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建两个实例标签，标签名为key1、key2，值为value1、value2。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchCreateOrDeleteRocketmqTagRequest()
        request.instance_id = "{instance_id}"
        listTagsbody = [
            TagEntity(
                key="key1",
                value="value1"
            ),
            TagEntity(
                key="key2",
                value="value2"
            )
        ]
        request.body = BatchCreateOrDeleteTagReq(
            tags=listTagsbody,
            action="create"
        )
        response = client.batch_create_or_delete_rocketmq_tag(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

创建两个实例标签，标签名为key1、key2，值为value1、value2。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchCreateOrDeleteRocketmqTagRequest{}
    request.InstanceId = "{instance_id}"
    keyTags:= "key1"
    valueTags:= "value1"
    keyTags1:= "key2"
    valueTags1:= "value2"
    var listTagsbody = []model.TagEntity{
        {
            Key: &keyTags,
            Value: &valueTags,
        },
        {
            Key: &keyTags1,
            Value: &valueTags1,
        },
    }
    actionBatchCreateOrDeleteTagReq:= model.GetBatchCreateOrDeleteTagReqActionEnum().CREATE
    request.Body = &model.BatchCreateOrDeleteTagReq{
        Tags: &listTagsbody,
        Action: &actionBatchCreateOrDeleteTagReq,
    }
    response, err := client.BatchCreateOrDeleteRocketmqTag(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
204	批量添加或删除实例标签成功。

错误码

请参见**错误码**。

5.8.2 查询实例标签

功能介绍

查询实例标签。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/rocketmq/{instance_id}/tags

表 5-159 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码： 200

表 5-160 响应 Body 参数

参数	参数类型	描述
tags	Array of **TagEntity** objects	标签列表

表 5-161 TagEntity

参数	参数类型	描述
key	String	标签键。 ● 不能为空。 ● 对于同一个实例，Key值唯一。 ● 长度为1~128个字符（中文也可以输入128个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ ● 首尾字符不能为空格。
value	String	标签值。 ● 长度为0~255个字符（中文也可以输入255个字符）。 ● 由任意语种字母、数字、空格和字符组成，字符仅支持_ . : = + - @ ● 首尾字符不能为空格。

请求示例

查询实例标签。

```
GET https://{endpoint}/v2/{project_id}/rocketmq/{instance_id}/tags
```

响应示例

状态码： 200

查询实例标签成功。

```
{
  "tags" : [ {
    "key" : "key1",
    "value" : "value1"
  }, {
    "key" : "key2",
    "value" : "value2"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
```

```
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowRocketmqTagsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowRocketmqTagsRequest request = new ShowRocketmqTagsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowRocketmqTagsResponse response = client.showRocketmqTags(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
```

```
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowRocketmqTagsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_rocketmq_tags(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowRocketmqTagsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowRocketmqTags(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询实例标签成功。

错误码

请参见**错误码**。

5.8.3 查询项目标签

功能介绍

查询项目标签。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/rocketmq/tags

表 5-162 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。

请求参数

无

响应参数

状态码： 200

表 5-163 响应 Body 参数

参数	参数类型	描述
tags	Array of **TagMultyValueEntity** objects	标签列表

表 5-164 TagMultyValueEntity

参数	参数类型	描述
key	String	标签键。
values	Array of strings	标签值。

请求示例

查询项目标签。

```
GET https://{endpoint}/v2/{project_id}/rocketmq/tags
```

响应示例

状态码： 200

查询项目标签成功。

```
{
  "tags" : [ {
    "key" : "key1",
    "values" : [ "value-test", "value1" ]
  }, {
    "key" : "key2",
    "values" : [ "value2" ]
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowRocketmqProjectTagsSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
```

```
            .withProjectId(projectId)
            .withAk(ak)
            .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
            .withCredential(auth)
            .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowRocketmqProjectTagsRequest request = new ShowRocketmqProjectTagsRequest();
        try {
            ShowRocketmqProjectTagsResponse response = client.showRocketmqProjectTags(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowRocketmqProjectTagsRequest()
        response = client.show_rocketmq_project_tags(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
```

```
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowRocketmqProjectTagsRequest{}
    response, err := client.ShowRocketmqProjectTags(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询项目标签成功。

错误码

请参见**错误码**。

5.9 其他接口

5.9.1 查询可用区信息

功能介绍

在创建实例时，需要配置实例所在的可用区ID，可通过该接口查询可用区的ID。

调用方法

请参见**如何调用API**。

URI

GET /v2/available-zones

请求参数

无

响应参数

状态码： 200

表 5-165 响应 Body 参数

参数	参数类型	描述
region_id	String	区域ID。
available_zones	Array of **available_zones** objects	可用区数组。

表 5-166 available_zones

参数	参数类型	描述
soldOut	Boolean	是否售罄。
id	String	可用区ID。
code	String	可用区编码。
name	String	可用区名称。
port	String	可用区端口号。
resource_availability	String	分区上是否还有可用资源。
default_az	Boolean	是否为默认可用区。
remain_time	Long	剩余时间。
ipv6_enable	Boolean	是否支持IPv6。

请求示例

查询可用区信息。

```
GET https://{endpoint}/v2/available-zones?engine=reliability
```

响应示例

状态码： 200

查询可用区信息成功。

```
{
  "region_id" : "xxx",
  "available_zones" : [ {
    "soldOut" : false,
    "id" : "8c90c2a4e2594c0782faa6b205afeca7",
    "code" : "xxx",
    "name" : "可用区1",
    "port" : "8002",
    "resource_availability" : "true",
    "default_az" : false,
    "remain_time" : 9223372036854776000,
    "ipv6_enable" : false
  }, {
    "soldOut" : false,
    "id" : "d539378ec1314c85b76fefa3f7071458",
    "code" : "xxx",
    "name" : "可用区2",
    "port" : "8003",
    "resource_availability" : "true",
    "default_az" : false,
    "remain_time" : 9223372036854776000,
    "ipv6_enable" : false
  }, {
    "soldOut" : false,
    "id" : "9f1c5806706d4c1fb0eb72f0a9b18c77",
    "code" : "xxx",
    "name" : "可用区3",
    "port" : "443",
    "resource_availability" : "true",
    "default_az" : true,
    "remain_time" : 9223372036854776000,
    "ipv6_enable" : false
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ListAvailableZonesSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
```

```
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");

        ICredential auth = new BasicCredentials()
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ListAvailableZonesRequest request = new ListAvailableZonesRequest();
        try {
            ListAvailableZonesResponse response = client.listAvailableZones(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")

    credentials = BasicCredentials(ak, sk) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListAvailableZonesRequest()
        response = client.list_available_zones(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
```

```
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListAvailableZonesRequest{}
    response, err := client.ListAvailableZones(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询可用区信息成功。

错误码

请参见**错误码**。

5.10 规格变更管理

5.10.1 查询实例的扩容规格列表

功能介绍

查询实例的扩容规格列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/{project_id}/instances/{instance_id}/extend

表 5-167 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎的类型。支持的类型为rocketmq。
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

表 5-168 Query 参数

参数	是否必选	参数类型	描述
type	否	String	产品的类型。 advanced：专享版

请求参数

无

响应参数

状态码： 200

表 5-169 响应 Body 参数

参数	参数类型	描述
engine	String	消息引擎类型
versions	Array of strings	消息引擎支持的版本
products	Array of **RocketMQExtendProductInfoEntity** objects	规格变更的产品信息

表 5-170 RocketMQExtendProductInfoEntity

参数	参数类型	描述
type	String	实例类型
product_id	String	产品ID
ecs_flavor_id	String	该产品使用的ECS规格
billing_code	String	账单计费类型。
arch_types	Array of strings	支持的CPU架构类型
charging_mode	Array of strings	支持的计费模式类型
ios	Array of **RocketMQExtendProductIosEntity** objects	磁盘IO信息
properties	**RocketMQExtendProductPropertiesEntity** object	功能特性的键值对
available_zones	Array of strings	有可用资源的可用区列表
unavailable_zones	Array of strings	资源售罄的可用区列表
support_features	Array of **RocketMQProductSupportFeaturesEntity** objects	支持的特性功能
qingtian_incompatible	Boolean	是否为擎天实例。

表 5-171 RocketMQExtendProductIosEntity

参数	参数类型	描述
io_spec	String	存储IO规格
available_zones	Array of strings	有可用资源的可用区列表
type	String	IO类型

参数	参数类型	描述
unavailable_zones	Array of strings	资源售罄的可用区列表

表 5-172 RocketMQExtendProductPropertiesEntity

参数	参数类型	描述
max_broker	String	Broker的最大个数。
max_topic_per_broker	String	每个节点最多能创建的Topic个数。
max_consumer_per_broker	String	每个节点的最大消费者数。
max_storage_per_node	String	每个节点的最大存储。单位为GB
min_broker	String	Broker的最小个数。
engine_versions	String	消息引擎版本。
min_storage_per_node	String	每个节点的最小存储。单位为GB
product_alias	String	product_id的别名

表 5-173 RocketMQProductSupportFeaturesEntity

参数	参数类型	描述
name	String	特性名称
properties	Map<String,String>	功能特性的键值对

请求示例

查询实例扩容列表。

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/extend
```

响应示例

状态码： 200

查询实例的扩容规格列表成功。

- {
  "engine" : "rocketmq",
  "versions" : ["4.8.0"],
  "products" : [{
    "type" : "cluster.small",
    "product_id" : "c6.4u8g.cluster.small",
    "ecs_flavor_id" : "c6.xlarge.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : ["X86"],
    "charging_mode" : ["monthly", "hourly"],
    "ios" : [{
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : ["xxx"],
      "type" : "evs",
      "unavailable_zones" : ["xxx"]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ ],
      "type" : "evs",
      "unavailable_zones" : ["xxx"]
    }],
    "support_features" : [ ],
    "properties" : {
      "max_broker" : "2",
      "max_topic_per_broker" : "2000",
      "max_storage_per_node" : "30000",
      "max_consumer_per_broker" : "2000",
      "min_broker" : "1",
      "product_alias" : "rocketmq.4u8g.cluster.small",
      "engine_versions" : "4.8.0",
      "min_storage_per_node" : "300"
    },
    "available_zones" : ["xxx"],
    "unavailable_zones" : ["xxx"],
    "qingtian_incompatible" : false
  }, {
    "type" : "cluster",
    "product_id" : "c6.4u8g.cluster",
    "ecs_flavor_id" : "s6.xlarge.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : ["X86"],
    "charging_mode" : ["monthly", "hourly"],
    "ios" : [{
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ ],
      "type" : "evs",
      "unavailable_zones" : ["xxx"]
    }, {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : ["xxx"],
      "type" : "evs",
      "unavailable_zones" : ["xxx"]
    }],
    "support_features" : [ ],
    "properties" : {
      "max_broker" : "10",
      "max_topic_per_broker" : "4000",
      "max_storage_per_node" : "60000",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "1",
      "product_alias" : "rocketmq.4u8g.cluster",
      "engine_versions" : "4.8.0",
      "min_storage_per_node" : "300"
    },
    "available_zones" : ["xxx"],
    "unavailable_zones" : ["xxx"],
    "qingtian_incompatible" : false
  }, {
    "type" : "cluster",


```
    "product_id" : "c6.8u16g.cluster",
    "ecs_flavor_id" : "c6s.2xlarge.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    } ],
    "support_features" : [ ],
    "properties" : {
      "max_broker" : "10",
      "max_topic_per_broker" : "8000",
      "max_storage_per_node" : "90000",
      "max_consumer_per_broker" : "8000",
      "min_broker" : "1",
      "product_alias" : "rocketmq.8u16g.cluster",
      "engine_versions" : "4.8.0",
      "min_storage_per_node" : "300"
    },
    "available_zones" : [ "xxx" ],
    "unavailable_zones" : [ "xxx" ],
    "qingtian_incompatible" : false
  }, {
    "type" : "cluster",
    "product_id" : "c6.12u24g.cluster",
    "ecs_flavor_id" : "c6s.3xlarge.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    } ],
    "support_features" : [ ],
    "properties" : {
      "max_broker" : "10",
      "max_topic_per_broker" : "12000",
      "max_storage_per_node" : "90000",
      "max_consumer_per_broker" : "12000",
      "min_broker" : "1",
      "product_alias" : "rocketmq.12u24g.cluster",
      "engine_versions" : "4.8.0",
      "min_storage_per_node" : "300"
    },
    "available_zones" : [ "xxx" ],
    "unavailable_zones" : [ "xxx" ],
    "qingtian_incompatible" : false
  }, {
    "type" : "cluster",
    "product_id" : "c6.16u32g.cluster",
    "ecs_flavor_id" : "c6.4xlarge.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
```

```
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ ],
      "type" : "evs",
      "unavailable_zones" : [ "xxx" ]
    } ],
    "support_features" : [ ],
    "properties" : {
      "max_broker" : "10",
      "max_topic_per_broker" : "16000",
      "max_storage_per_node" : "90000",
      "max_consumer_per_broker" : "16000",
      "min_broker" : "1",
      "product_alias" : "rocketmq.16u32g.cluster",
      "engine_versions" : "4.8.0",
      "min_storage_per_node" : "300"
    },
    "available_zones" : [ "xxx" ],
    "unavailable_zones" : [ "xxx" ],
    "qingtian_incompatible" : false
  } ]
}
```


SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ShowEngineInstanceExtendProductInfoSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowEngineInstanceExtendProductInfoRequest request = new
```

```
ShowEngineInstanceExtendProductInfoRequest();
        request.withEngine("{engine}");
        request.withInstanceId("{instance_id}");
        try {
            ShowEngineInstanceExtendProductInfoResponse response =
client.showEngineInstanceExtendProductInfo(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security
risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment
variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this
example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowEngineInstanceExtendProductInfoRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        response = client.show_engine_instance_extend_product_info(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
```

```
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowEngineInstanceExtendProductInfoRequest{}
    request.Engine = "{engine}"
    request.InstanceId = "{instance_id}"
    response, err := client.ShowEngineInstanceExtendProductInfo(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	查询实例的扩容规格列表成功。

错误码

请参见**错误码**。

5.10.2 实例规格变更

功能介绍

实例规格变更。

当前通过调用API，只支持按需实例进行实例规格变更。

调用方法

请参见**如何调用API**。

URI

POST /v2/{engine}/{project_id}/instances/{instance_id}/extend

表 5-174 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎的类型。支持的类型为rocketmq。
project_id	是	String	项目ID，获取方式请参见**获取项目ID**。
instance_id	是	String	实例ID。

请求参数

表 5-175 请求 Body 参数

参数	是否必选	参数类型	描述
oper_type	是	String	变更类型 取值范围： • storage：存储空间扩容，代理数量不变。 • horizontal：代理数量扩容，每个broker的存储空间不变。 • vertical：垂直扩容，broker的底层虚机规格变更，代理数量和存储空间不变。
new_storage_space	否	Integer	当oper_type类型是storage或horizontal时，该参数有效且必填，实例存储空间 = 代理数量 * 每个broker的存储空间。 • 当oper_type类型是storage时，代理数量不变，每个broker存储空间最少扩容100GB。 • 当oper_type类型是horizontal时，每个broker的存储空间不变。

参数	是否必选	参数类型	描述
new_product_id	否	String	当oper_type类型是vertical时，该参数才有效且必填。
new_broker_num	否	Integer	代理数量 当oper_type参数为horizontal时，该参数必填。
publicip_id	否	String	实例绑定的弹性IP地址的ID。以英文逗号隔开多个弹性IP地址的ID。 当oper_type参数为horizontal且开启了公网访问时，此参数必填。

响应参数

状态码： 200

表 5-176 响应 Body 参数

参数	参数类型	描述
job_id	String	规格变更任务ID。

请求示例

```
POST https://{endpoint}/v2/rocketmq/{project_id}/instances/{instance_id}/extend

{
  "oper_type" : "horizontal",
  "new_product_id" : "c6.4u8g.cluster"
}
```

响应示例

状态码： 200

实例规格变更成功。

```
{
  "job_id" : "93b94287-728d-4bb1-a158-cb66cb0854e7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

```
package com.huaweicloud.sdk.test;

import com.huaweicloud.sdk.core.auth.ICredential;
import com.huaweicloud.sdk.core.auth.BasicCredentials;
```

```
import com.huaweicloud.sdk.core.exception.ConnectionException;
import com.huaweicloud.sdk.core.exception.RequestTimeoutException;
import com.huaweicloud.sdk.core.exception.ServiceResponseException;
import com.huaweicloud.sdk.rocketmq.v2.region.rocketmqRegion;
import com.huaweicloud.sdk.rocketmq.v2.*;
import com.huaweicloud.sdk.rocketmq.v2.model.*;


public class ResizeInstanceSolution {

    public static void main(String[] args) {
        // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
        // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
        String ak = System.getenv("CLOUD_SDK_AK");
        String sk = System.getenv("CLOUD_SDK_SK");
        String projectId = "{project_id}";

        ICredential auth = new BasicCredentials()
                .withProjectId(projectId)
                .withAk(ak)
                .withSk(sk);

        rocketmqClient client = rocketmqClient.newBuilder()
                .withCredential(auth)
                .withRegion(rocketmqRegion.valueOf("<YOUR REGION>"))
                .build();
        ResizeInstanceRequest request = new ResizeInstanceRequest();
        request.withEngine("{engine}");
        request.withInstanceId("{instance_id}");
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        body.withNewProductId("c6.4u8g.cluster");
        body.withOperType("horizontal");
        request.withBody(body);
        try {
            ResizeInstanceResponse response = client.resizeInstance(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
            System.out.println(e.getHttpStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrocketmq.v2.region.rocketmq_region import rocketmqRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrocketmq.v2 import *

if __name__ == "__main__":
    # The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    # In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak = __import__('os').getenv("CLOUD_SDK_AK")
```

```
    sk = __import__('os').getenv("CLOUD_SDK_SK")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId) \

    client = rocketmqClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(rocketmqRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeInstanceRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.body = ResizeEngineInstanceReq(
            new_product_id="c6.4u8g.cluster",
            oper_type="horizontal"
        )
        response = client.resize_instance(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

Go

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rocketmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rocketmq/v2/region"
)

func main() {
    // The AK and SK used for authentication are hard-coded or stored in plaintext, which has great security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.
    // In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment
    ak := os.Getenv("CLOUD_SDK_AK")
    sk := os.Getenv("CLOUD_SDK_SK")
    projectId := "{project_id}"

    auth := basic.NewCredentialsBuilder().
        WithAk(ak).
        WithSk(sk).
        WithProjectId(projectId).
        Build()

    client := rocketmq.NewrocketmqClient(
        rocketmq.rocketmqClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResizeInstanceRequest{}
    request.Engine = "{engine}"
    request.InstanceId = "{instance_id}"
    newProductIdResizeEngineInstanceReq:= "c6.4u8g.cluster"
    request.Body = &model.ResizeEngineInstanceReq{
        NewProductId: &newProductIdResizeEngineInstanceReq,
        OperType: "horizontal",
    }
    response, err := client.ResizeInstance(request)
```

```
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

更多

更多编程语言的SDK代码示例，请参见**API Explorer**的代码示例页签，可生成自动对应的SDK代码示例。

状态码

状态码	描述
200	实例规格变更成功。

错误码

请参见**错误码**。

6 权限和授权项

如果您需要对您所拥有的DMS for RocketMQ实例进行精细的权限管理，您可以使用统一身份认证服务（Identity and Access Management，简称IAM），如果华为账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用DMS for RocketMQ实例的其它功能。

默认情况下，新建的IAM用户没有任何权限，您需要将其加入用户组，并给用户组授予策略或角色，才能使用户组中的用户获得相应的权限，这一过程称为授权。授权后，用户就可以基于已有权限对云服务进行操作。

权限根据授权的精细程度，分为**角色**和**策略**。角色以服务为粒度，是IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。策略以API接口为粒度进行权限拆分，授权更加精细，可以精确到某个操作、资源和条件，能够满足企业对权限最小化的安全管控要求。

说明

如果您要允许或是禁止某个接口的操作权限，请使用策略。

账号具备所有接口的调用权限，如果使用账号下的IAM用户发起API请求时，该IAM用户必须具备调用该接口所需的权限，否则，API请求将调用失败。每个接口所需要的权限，与各个接口所对应的授权项相对应，只有发起请求的用户被授予授权项所对应的策略，该用户才能成功调用该接口。例如，用户要调用接口来创建RocketMQ实例，那么这个IAM用户被授予的策略中必须包含允许“dms:instance:create”的授权项，该接口才能调用成功。

支持的授权项

策略包含系统策略和自定义策略，如果系统策略不满足授权要求，管理员可以创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制。策略支持的操作与API相对应，授权项列表说明如下：

- 权限：允许或拒绝某项操作。
- 对应API接口：自定义策略实际调用的API接口。
- 授权项：自定义策略中支持的Action，在自定义策略中的Action中写入授权项，可以实现授权项对应的权限功能。
- IAM项目（Project）/企业项目（Enterprise Project）：自定义策略的授权范围，包括IAM项目与企业项目。授权范围如果同时支持IAM项目和企业项目，表示此授权项对应的自定义策略，可以在IAM和企业管理两个服务中给用户组授权并生

效。如果仅支持IAM项目，不支持企业项目，表示仅能在IAM中给用户组授权并生效，如果在企业管理中授权，则该自定义策略不生效。关于IAM项目与企业项目的区别，详情请参见：**IAM与企业管理的区别**。

DMS for RocketMQ的支持自定义策略授权项如下表所示。用户调用如下API时，需要获取对应的权限。权限获取请参考**统一身份认证服务（IAM）**的帮助指导。

表 6-1 DMS for RocketMQ 授权项明细

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
创建实例（按需）	POST /v2/{project_id}/instances	dms:instance:create	√	√
查询所有实例列表	GET /v2/{project_id}/instances	dms:instance:list	√	√
查询指定实例	GET /v2/{project_id}/instances/{instance_id}	dms:instance:get	√	√
删除指定的实例	DELETE /v2/{project_id}/instances/{instance_id}	dms:instance:delete	√	√
修改实例信息	PUT /v2/{project_id}/instances/{instance_id}	dms:instance:modify	√	√
批量删除实例	POST /v2/{project_id}/instances/action	dms:instance:delete	√	√
修改主题	PUT /v2/{project_id}/instances/{instance_id}/topics/{topic}	dms:instance:modify	√	√
查询主题消费组列表	GET /v2/{project_id}/instances/{instance_id}/topics/{topic}/groups	dms:instance:get	√	√
查询主题的消息数	GET /v2/{project_id}/instances/{instance_id}/topics/{topic}/status	dms:instance:get	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询消息	GET /v2/{engine}/{project_id}/instances/{instance_id}/messages	dms:instance:get	√	√
创建用户	POST /v2/{project_id}/instances/{instance_id}/users	dms:instance:modify	√	√
查询用户列表	GET /v2/{project_id}/instances/{instance_id}/users	dms:instance:get	√	√
查询用户详情	GET /v2/{project_id}/instances/{instance_id}/users/{user_name}	dms:instance:get	√	√

7 附录

7.1 状态码

状态码如表7-1所示

表 7-1 状态码

状态码	编码	错误码说明
100	Continue	继续请求。 这个临时响应用来通知客户端，它的部分请求已经被服务器接收，且仍未被拒绝。
101	Switching Protocols	切换协议。只能切换到更高级的协议。 例如，切换到HTTP的新版本协议。
200	OK	请求成功。
201	Created	创建类的请求完全成功。
202	Accepted	已经接受请求，但未处理完成。
203	Non-Authoritative Information	非授权信息，请求成功。
204	NoContent	请求完全成功，同时HTTP响应不包含响应体。 在响应OPTIONS方法的HTTP请求时返回此状态码。
205	Reset Content	重置内容，服务器处理成功。
206	Partial Content	服务器成功处理了部分GET请求。
300	Multiple Choices	多种选择。请求的资源可包括多个位置，相应可返回一个资源特征与地址的列表用于用户终端（例如：浏览器）选择。

状态码	编码	错误码说明
301	Moved Permanently	永久移动，请求的资源已被永久的移动到新的URI，返回信息会包括新的URI。
302	Found	资源被临时移动。
303	See Other	查看其它地址。 使用GET和POST请求查看。
304	Not Modified	所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
305	Use Proxy	所请求的资源必须通过代理访问。
306	Unused	已经被废弃的HTTP状态码。
400	BadRequest	非法请求。 建议直接修改该请求，不要重试该请求。
401	Unauthorized	在客户端提供认证信息后，返回该状态码，表明服务端指出客户端所提供的认证信息不正确或非法。
402	Payment Required	保留请求。
403	Forbidden	请求被拒绝访问。 返回该状态码，表明请求能够到达服务端，且服务端能够理解用户请求，但是拒绝做更多的事情，因为该请求被设置为拒绝访问，建议直接修改该请求，不要重试该请求。
404	NotFound	所请求的资源不存在。 建议直接修改该请求，不要重试该请求。
405	MethodNotAllowed	请求中带有该资源不支持的方法。 建议直接修改该请求，不要重试该请求。
406	Not Acceptable	服务器无法根据客户端请求的内容特性完成请求。
407	Proxy Authentication Required	请求要求代理的身份认证，与401类似，但请求者应当使用代理进行授权。
408	Request Time-out	服务器等候请求时发生超时。 客户端可以随时再次提交该请求而无需进行任何更改。
409	Conflict	服务器在完成请求时发生冲突。 返回该状态码，表明客户端尝试创建的资源已经存在，或者由于冲突请求的更新操作不能被完成。
410	Gone	客户端请求的资源已经不存在。 返回该状态码，表明请求的资源已被永久删除。

状态码	编码	错误码说明
411	Length Required	服务器无法处理客户端发送的不带Content-Length的请求信息。
412	Precondition Failed	未满足前提条件，服务器未满足请求者在请求中设置的其中一个前提条件。
413	Request Entity Too Large	由于请求的实体过大，服务器无法处理，因此拒绝请求。为防止客户端的连续请求，服务器可能会关闭连接。如果只是服务器暂时无法处理，则会包含一个Retry-After的响应信息。
414	Request-URI Too Large	请求的URI过长（URI通常为网址），服务器无法处理。
415	Unsupported Media Type	服务器无法处理请求附带的媒体格式。
416	Requested range not satisfiable	客户端请求的范围无效。
417	Expectation Failed	服务器无法满足Expect的请求头信息。
422	UnprocessableEntity	请求格式正确，但是由于含有语义错误，无法响应。
429	TooManyRequests	表明请求超出了客户端访问频率的限制或者服务端接收到多于它能处理的请求。建议客户端读取相应的Retry-After首部，然后等待该首部指出的时间后再重试。
500	InternalServerError	表明服务端能被请求访问到，但是不能理解用户的请求。
501	Not Implemented	服务器不支持请求的功能，无法完成请求。
502	Bad Gateway	充当网关或代理的服务器，从远端服务器接收到了一个无效的请求。
503	ServiceUnavailable	被请求的服务无效。 建议直接修改该请求，不要重试该请求。
504	ServerTimeout	请求在给定的时间内无法完成。客户端仅在为请求指定超时（Timeout）参数时会得到该响应。
505	HTTP Version not supported	服务器不支持请求的HTTP协议的版本，无法完成处理。

7.2 错误码

当您调用API时，如果遇到“APIGW”开头的错误码，请参见**API网关错误码**进行处理。

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400002	The project ID format is invalid.	Project-ID的格式无效。	请检查Project-ID的格式
400	DMS.00400004	The request body is empty.	请求消息体为空。	请查看请求信息体
400	DMS.00400005	The message body is not in JSON format or contains invalid characters.	请求消息体不是JSON格式或字段非法。	请检查消息体格式
400	DMS.00400007	Unsupported type.	不支持的类型。	请检查类型
400	DMS.00400008	Unsupported version.	不支持的版本。	请检查版本
400	DMS.00400009	Invalid product_id.	请求参数product_id非法。	请检查参数product_id
400	DMS.00400010	Invalid instance name. The name must be 4 to 64 characters long. Only letters, digits, underscores (_), and hyphens (-) are allowed.	实例名称不合法，只能包含字母，数字，下划线或者中划线，长度为4-64。	请检查实例名称
400	DMS.00400011	The instance description can contain a maximum of 1024 characters.	实例描述长度必须为0-1024。	请查看实例描述

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400012	The password does not meet the complexity requirements. An instance password must:Be a string consisting of 8 to 32 characters.Contain at least two of the following character types:Lowercase lettersUppercase lettersDigitsSpecial characters `~!@#$%^&*()-_=+|[{}];:',<.>/?	密码格式不符合要求。 密码复杂度要求：1、输入长度为8到32位的字符串。 2、必须包含如下四种字符中的三种组合：小写字母、大写字母、数字、特殊字符包括（`~!@#$%^&*()-_=+|[{}]:'",./?）3、不能与校验的弱密码相同。	请确认密码是否符合要求
400	DMS.00400013	vpc_id in the request is empty.	请求参数vpc_id为空。	请检查参数vpc_id
400	DMS.00400014	security_group_id in the request is empty.	请求参数security_group_id为空。	请检查参数security_group_id
400	DMS.00400015	Invalid username. A username must be 4 to 64 characters long and consist of only letters, digits, and hyphens (-).	用户名不符合要求，用户名只能由英文字母、数字、中划线组成，长度为4~64的字符。	请检查用户名
400	DMS.00400016	subnet_id in the request is empty.	请求参数subnet_id为空。	请检查参数subnet_id

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400017	This DMS instance job task is still running.	实例任务状态运行中	请稍后再试
400	DMS.00400018	This subnet must exist in the VPC.	子网必须在VPC中存在。	请检查子网
400	DMS.00400019	The password does not meet the complexity requirements.	密码复杂度不符合要求。	请检查密码复杂度
400	DMS.00400020	DHCP must be enabled for this subnet.	子网的DHCP必须开启。	请检查DHCP
400	DMS.00400021	The isAutoRenew parameter in the request must be either 0 or 1.	请求参数isAutoRenew非法。	请检查参数isAutoRenew
400	DMS.00400022	Engine does not match the product id.	Engine和ProductID不匹配。	请检查参数engine
400	DMS.00400026	This operation is not allowed due to the instance status.	当前的实例状态不支持该操作。	请检查实例状态
400	DMS.00400028	Query advanced product, specCode not exists.	查询高级特性product specCode不存在。	请检查参数origin_spec_code。
400	DMS.00400029	Query advanced product failed, can not find product for request.	查询高级特性product specCode不存在。	请检查参数origin_spec_code。

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400030	Invalid DMS instance id. The id must be a uuid.	实例id不合法。	请检查参数id。
400	DMS.00400035	DMS instance quota of the tenant is insufficient.	租户实例配额不足。	请申请扩大配额。
400	DMS.00400037	The instanceParams parameter in the request contains invalid characters or is not in JSON format.	请求参数instanceParams非法，不是JSON格式或字段非法。	请检查请求参数
400	DMS.00400038	The periodNum parameter in the request must be an integer.	请求参数periodNum非法，必须为整数。	请检查参数periodNum
400	DMS.00400039	The quota limit has been reached.	请求调整配额超出限制范围。	请申请扩大配额。
400	DMS.00400042	The AZ does not exist.	可用区不存在。	请检查可用区
400	DMS.00400045	The instance is not frozen and cannot be unfrozen.	实例没有被冻结，不能进行解除冻结操作。	请查询实例状态
400	DMS.00400046	This security group does not exist.	安全组不存在。	请检查安全组
400	DMS.00400047	The periodType parameter in the request must be either 2 or 3.	请求参数periodType非法。	请检查参数periodType

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400048	Invalid security group rules. Ensure that rules with the protocol being ANY are configured for both the inbound and outbound directions.	安全组规则不符合要求，请确保安全组规则中同时包含协议为“ANY”的出方向和入方向规则。	请检查安全组规则
400	DMS.00400049	The availability zone does not support ipv6.	可用分区不支持IPv6。	请重新选择可用分区
400	DMS.00400051	not found the new setup version tar to upgrade instance.	实例升级未找到新版本安装包。	请重新选择升级的版本号
400	DMS.00400052	only the instance at running status can upgrade.	实例升级状态必须是RUNNING。	请稍后再试
400	DMS.00400053	the upgrade instance version equals to current version.	升级版本与当前版本相同	请重新选择升级的版本号
400	DMS.00400055	Resource sold out.	资源不足，包括ecs，volume等	请稍后再试
400	DMS.00400060	This instance name already exists.	实例名称已经存在。	请检查实例名称
400	DMS.00400061	Invalid instance ID format.	实例ID的格式无效。	请检查实例ID
400	DMS.00400062	Invalid request parameter.	请求参数无效	请检查请求参数

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400063	Invalid configuration parameter {0}.	配置参数{0}非法。	请检查参数
400	DMS.00400064	The action parameter in the request must be delete or restart.	请求参数action非法，只能为delete或restart。	请检查参数action
400	DMS.00400065	The instances parameter in the request is empty.	请求参数instances为空。	请检查参数instances
400	DMS.00400066	Invalid configuration parameter {0}.	配置参数{0}非法。	请检查参数
400	DMS.00400067	The available_zones parameter in the request must be an array that contains only one AZ ID.	请求参数available_zones非法，必须为只包含一个可用区ID的数组。	请检查参数available_zones
400	DMS.00400068	The VPC does not exist.	VPC不存在。	请检查VPC
400	DMS.00400070	Invalid task ID format.	任务ID的格式无效。	请检查任务ID
400	DMS.00400081	Duplicate instance name.	实例名称重复。	请检查实例名称
400	DMS.00400082	Instance id is repeated.	实例ID重复。	请检查实例ID
400	DMS.00400085	The message body contains invalid characters or is not in JSON format. The error key is <key>.	请求消息体不是JSON格式或字段非法，有明确的错误字段。	请检查错误字段

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400099	The following instances in the Creating, Starting, Stopping, or Restarting state cannot be deleted.	实例状态为创建中、启动中、停止中、重启中时不允许执行删除操作。错误的实例为：{}	请检查实例状态
400	DMS.00400100	The instances array can contain a maximum of 50 instance IDs.	instances数组最多只能包含50个实例ID。	请检查实例数量
400	DMS.00400101	The name of a Kafka topic must be 4 to 64 characters long and start with a letter. Only letters, digits, underscores (_), and hyphens (-) are allowed.	Kafka实例创建Topic的名称必须以字母开头且只支持大小写字母、中横线、下划线以及数字，长度为4-64。	请检查topic名称
400	DMS.00400102	The number of partitions created for a Kafka topic must be within the range of 1-20.	Kafka实例创建Topic的分区数必须在1-20范围内。	请检查topic分区数
400	DMS.00400103	The number of replicas created for a Kafka topic must be within the range of 1-20.	Kafka实例创建Topic的副本数必须在1-20范围内。	请检查topic副本数

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400105	The message retention period of a Kafka topic must be within the range of 1-168.	Kafka实例创建Topic的老化时间必须在1-168范围内。	请检查Topic老化时间
400	DMS.00400106	Invalid maintenance time window.	维护时间窗参数非法。	请检查维护时间窗参数
400	DMS.00400107	The instance exists for unpaid sacle up orders. Please process non payment orders first.	该实例的扩容订单已存在，请先处理订单。	请处理已存在的订单
400	DMS.00400108	The Instance exists for processing sacle up order. Please try again later.	该实例的扩容订单正在处理中，请稍后重试。	请稍后再试
400	DMS.00400124	The maximum number of disk expansion times has been reached.	超过磁盘最大扩容次数	请检查磁盘最大扩容次数
400	DMS.00400125	Invalid SPEC_CODE.	SPEC_CODE不合法	请检查SPEC_CODE
400	DMS.00400126	Invalid period time.	无效的包周期时间。	请检查包周期时间
400	DMS.00400127	Instance not support to change retention_policy.	实例不支持修改老化策略。	请联系技术支持
400	DMS.00400128	Invalid public access parameters.	公网访问参数错误。	请检查公网访问参数

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400129	Current instance version is less than required.	当前版本的实例不支持该操作。	请联系技术支持
400	DMS.00400133	Sink task quota for connector invalid.	无效的connector任务配额。	请联系技术支持
400	DMS.00400134	There is another order need to pay first.	已存在未支付的订单。	请继续支付订单
400	DMS.00400135	Not support disk encrypted.	不支持磁盘加密。	请不要选用磁盘加密功能
400	DMS.00400136	Disk encrypted key is null.	磁盘加密的密钥是空值。	请检查磁盘加密的密钥
400	DMS.00400137	Disk encrypted key state is not enabled.	磁盘加密密钥状态不是开启。	请开启磁盘加密状态
400	DMS.00400142	Timestamp is invalid.	时间戳无效。	请输入正确的时间戳。
400	DMS.00400500	Invalid disk space.	磁盘空间不合法	请检查磁盘空间
400	DMS.00400800	Invalid request parameter. Check the request parameter.	请求参数不合法	请检查请求参数
400	DMS.00400861	Replication factor larger than available brokers.	创建Topic的副本数大于当前可用的Broker数。	请联系技术支持工程师协助解决。
400	DMS.00400867	Failed to create the Smart Connect task.	创建connector task失败	请联系技术支持。
400	DMS.00400868	Failed to stop the Smart Connect task.	停止connector task失败	请稍后再试。

状态码	错误码	错误信息	描述	处理措施
400	DMS.00400869	Failed to start the Smart Connect task.	启动connector task失败	请稍后再试。
400	DMS.00400870	Failed to verify the Smart Connect task.	校验connector task失败	请稍后再试。
400	DMS.00400872	Failed to restart the Smart Connect task.	重启connector task失败	请稍后再试。
400	DMS.00400873	Failed to modify the Smart Connect task.	修改connector task参数失败	请联系技术支持。
400	DMS.00400874	The topic has been used in another Smart Connect task.	Topic已经在其他SmartConnectTask中使用	请核实Topic后重试。
400	DMS.00400875	Inconsistent source and target Redis instance types in the Smart Connect task.	connector task 源端和目的端redis类型不符	请修改源端和目的端redis类型后重试。
400	DMS.00400876	The topic does not exist.	Topic不存在	请核实Topic后重试。
400	DMS.00400970	RabbitMQ plugin is not exist	插件名称非法	请检查插件名称是否在插件列表中
400	DMS.00400971	The instance ssl is off.	实例未开启ssl。	请检视实例详情，是否开启ssl。
400	DMS.00400975	Failed to query topics.	查询topic失败。	请检查topic是否存在。
400	DMS.00404033	Does not support extend rabbitMQ disk space.	不支持扩容RabbitMQ的磁盘空间。	请使用扩大集群的方式扩容RabbitMQ实例。
400	DMS.00500960	Invalid user AK/SK.	用户AK SK非法	请核实用户AK SK后重试。

状态码	错误码	错误信息	描述	处理措施
400	DMS.00500986	Your account has been restricted.	您的账户被限制	联系计费中心处理
400	DMS.00500987	Balance is not enough	余额不足	请充值后重试
400	DMS.10240002	The number of queried queues exceeds the upper limit.	查询队列的数量超过了范围。	请检查队列数量
400	DMS.10240004	The tag name is invalid.	Tag名称无效。	请检查tag名称
400	DMS.10240005	The project ID format is invalid.	Project ID的格式无效。	请检查projectid格式
400	DMS.10240007	The name contains invalid characters.	名称包含无效字符。	请检查名称
400	DMS.10240009	The message body is not in JSON format or contains invalid characters.	消息体不是JSON格式或字段非法。	请检查消息体
400	DMS.10240010	The description contains invalid characters.	描述包含无效字符。	请检查描述
400	DMS.10240011	The name length must be 1 to 64 characters.	名称长度必须为[1,64]。	请检查名称长度
400	DMS.10240012	The name length must be 1 to 32 characters.	名称长度必须为[1,32]。	请检查名称长度

状态码	错误码	错误信息	描述	处理措施
400	DMS.10240013	The description length must not exceed 160 characters.	描述长度必须为[0,160]。	请检查描述长度
400	DMS.10240014	The number of consumable messages exceeds the maximum limit.	最大消费消息数不在合法范围内。	请检查最大消费消息数量
400	DMS.10240015	The queue ID format is invalid.	Queue ID的格式无效。	请检查queueid
400	DMS.10240016	The group ID format is invalid.	Group ID的格式无效。	请检查groupid
400	DMS.10240017	The queue already exists.	队列已经存在。	请检查队列是否已存在
400	DMS.10240018	The consumer group already exists.	消费组已存在。	请检查消费组是否已存在
400	DMS.10240019	The number of consumer groups exceeds the upper limit.	消费组的数目超出限制。	请检查消费组数量
400	DMS.10240020	The quota is insufficient.	配额不足。	请检查配额
400	DMS.10240021	The value of time_wait is not within the value range of 1-60.	消费等待时间不在[1,60]范围内。	请检查消费等待时间
400	DMS.10240022	The value of max Consume Count must be within the range of 1-100.	进入死信队列前的最大消费次数的值必须在[1,100]范围内。	请检查死信队列最大消费次数的值

状态码	错误码	错误信息	描述	处理措施
400	DMS.10240027	The value of retention_hours must be an integer in the range of 1-72.	Kafka队列的消息保存时间必须在[1,72]范围内。	请检查kafka队列消息保存时间
400	DMS.10240028	Non-kafka queues do not support retention_hours.	非Kafka队列不能设置消息保存时间。	请检查是否是kafka队列，如果不是请不要设置消息保存时间
400	DMS.10240032	The queue is being created.	队列正在创建。	请检查队列是否已在创建中
400	DMS.10240035	The tag key is empty or too long.	队列标签的键不能为空，或者长度太长。	请检查队列标签的键
400	DMS.10240036	The tag key contains invalid characters.	队列标签的键包含非法字符。	请检查队列标签的键
400	DMS.10240038	The tag value is too long.	队列标签的值太长。	请检查队列标签的值
400	DMS.10240039	The tag value contains invalid characters.	标签的值包含非法字符。	请检查队列标签的值
400	DMS.10240040	You can only create or delete tags.	只能支持创建或者删除的操作。	请检查操作是否符合要求
400	DMS.10240041	You can only filter or count tags.	只能支持过滤或者统计的操作。	请检查操作是否符合要求
400	DMS.10240042	The number of records on each page for pagination query exceeds the upper limit.	分页查找的分页大小超出范围。	请检查分页大小

状态码	错误码	错误信息	描述	处理措施
400	DMS.10240043	The number of skipped records for pagination query exceeds the upper limit.	分页查找的分页偏移超出范围。	请检查分页偏移
400	DMS.10240044	A maximum of 10 tags can be created.	不能创建超过10个标签。	请检查标签数量
400	DMS.10240045	The tag key has been used.	标签的键已经被使用过。	请检查标签是否已使用
400	DMS.10540001	The message body contains invalid fields.	消息体的字段非法。	请检查消息体字段
400	DMS.10540003	Message ack status must be either 'success' or 'fail'. It should not be '{status}'.	消息确认status字段值必须为'success'或'fail'，目前为{status}。	请检查请求字段status是否不符合要求
400	DMS.10540004	Request error	请求错误：queue或group name与handler的信息不匹配。	请检查queue或group name与handler的信息
400	DMS.10540010	The request format is incorrect	请求的格式错误：{错误描述信息}。	请检查请求格式
400	DMS.10540011	The message size is {message size}, larger than the size limit {max allowed size}.	请求消息大小超过阈值，目前为{消息大小}，最大限制为：{最大消息大小}。	请检查请求消息大小
400	DMS.10540012	The message body is not in JSON format or contains invalid characters.	消息体不是JSON格式或字段非法。	请检查消息体格式

状态码	错误码	错误信息	描述	处理措施
400	DMS.10540014	The URL contains invalid parameters.	URI中参数错误。	请检查url参数
400	DMS.10540202	The request format is incorrect	请求的格式错误：{错误描述信息} 。	请检查请求格式
400	DMS.10542204	Failed to consume messages due to {desc}.	消费消息失败，错误信息为：{错误描述}。	请查看错误信息并做对应处理
400	DMS.10542205	Failed to obtain the consumption instance because the handler does not exist. This may be because the consumer instance is released 1 minute after the message is consumed. As a result, the consumer instance fails to be obtained from the handler.	handler不存在，获取消费实例失败，可能是因为1分钟后消费实例释放，导致从handler获取consumer实例失败。	请检查handler
400	DMS.10542206	The value of ack_wait must be within the range of 15-300.	ack_wait的取值必须在15~300范围内。	请检查ack_wait的取值

状态码	错误码	错误信息	描述	处理措施
400	DMS.10542209	The handler does not exist because the handler fails to be parsed, the message consumption times out, or the message consumption is repeatedly acknowledged.	handler不存在，可能是因为handler解析失败、消费确认超时或重复确认。	请检查handler或者消费确认是否超时
400	DMS.10542214	The request format is incorrect	请求的格式错误：{错误描述信息}。	请检查请求格式
400	DMS.50050004	The consumer group is offline.	消费组不在线	启动该消费组内消费者实例
401	DMS.10240101	Invalid token.	Token无效。	请检查token是否有效
401	DMS.10240102	Expired token.	Token已过期。	请检查使用的token是否已过期
401	DMS.10240103	Missing token.	Token缺失。	请检查是否没有token
401	DMS.10240104	The project ID and token do not match.	Project-ID和Token不匹配。	请检查projectid和token是否匹配
403	DMS.00403002	A tenant has the read-only permission and cannot perform operations on DMS.	租户只有只读权限，无法操作DMS。	请检查租户权限
403	DMS.00403003	This role does not have the permissions to perform this operation.	角色没有操作权限，无法执行此操作。	请检查角色权限

状态码	错误码	错误信息	描述	处理措施
403	DMS.10240304	Change the quota of a queue or consumer group to a value smaller than the used quota.	修改队列或者消费组的配额小于已使用的数量。	请检查配额
403	DMS.10240306	The tenant has been frozen. You cannot perform operations on DMS.	租户已被冻结，您无法操作DMS消息队列服务。	请检查租户状态
403	DMS.10240307	The consumer group quota must be within the range of 1-10.	消费组的配额必须在[1,10]范围内。	请检查消费组数量是否已超过配额
403	DMS.10240308	The queue quota must be within the range of 1-20.	队列的配额必须在[1,20]范围内。	请检查队列数量是否已超过配额
403	DMS.10240309	Access denied. You cannot perform operations on DMS.	访问被拒绝，您无法操作DMS消息队列服务。	请检查是否有操作DMS权限
403	DMS.10240310	A tenant has the read-only permission and cannot perform operations on DMS.	租户只读权限，您无法操作DMS消息队列服务。	请检查租户权限
403	DMS.10240311	This role does not have the permissions to perform this operation.	角色没有操作权限，您无法操作DMS消息队列服务。	请检查角色权限

状态码	错误码	错误信息	描述	处理措施
403	DMS.10240312	The tenant is restricted and cannot perform operations on DMS.	租户受限，您无法操作DMS消息队列服务。	请检查角色权限
404	DMS.00404001	The requested URL does not exist.	请求的URL不存在。	请检查url
404	DMS.00404022	This instance does not exist.	实例不存在。	请检查是否存在该实例
404	DMS.00404024	Connector does not exist.	Connector不存在。	请检查Connector
404	DMS.00404026	The dumping task does not exist.	转储任务不存在。	请检查转储任务
404	DMS.00404027	Connector already exists.	Connector已存在。	请检查Connector
404	DMS.00404029	The dumping task quota has been reached.	超过最大转储任务配额。	请检查转储任务配额
404	DMS.10240401	The queue ID is incorrect or not found.	队列ID错误或者没找到。	请检查对应的队列ID是否存在且正确
404	DMS.10240405	The consumption group ID is incorrect or not found.	消费组ID错误或者没找到。	请检查对应的消费组ID是否存在且正确
404	DMS.10240406	The URL or endpoint does not exist.	Url或Endpoint不存在。	请检查对应的Url或Endpoint是否存在且正确
404	DMS.10240407	The request is too frequent. Flow control is being performed. Please try again later.	请求过于频繁，正在流控，请稍后再试。	请稍后再试

状态码	错误码	错误信息	描述	处理措施
404	DMS.10240426	No tag containing this key exists.	不存在包含该键的标签。	请检查标签
404	DMS.10540401	The queue name does not exist.	队列名不存在。	请检查队列名是否存在
405	DMS.00405001	This request method is not allowed.	请求中指定的方法不被允许。	请检查请求方法
408	DMS.111501024	Query timed out	消息查询超时	请稍后查询
500	DMS.00500000	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500006	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500017	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500024	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500025	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500041	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500052	Internal service error.	实例升级JOB提交失败。	请联系技术支持。
500	DMS.00500053	Internal service error.	未找到实例节点。	请联系技术支持。
500	DMS.00500054	Internal service error.	生成密码错误。	请联系技术支持。
500	DMS.00500070	Internal service error.	实例配置失败。	请联系技术支持。
500	DMS.00500071	Internal service error.	创建实例备份策略失败。	请联系技术支持。
500	DMS.00500094	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500106	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.00500990	Failed to update topics.	更新topic失败。	请联系技术支持。

状态码	错误码	错误信息	描述	处理措施
500	DMS.00501000	Failed to create agency, may be you do not have the agency permission.	创建委托失败	检查当前用户是否有委托权限
500	DMS.00501001	Failed to get agency roleId.	移除委托的权限策略失败	请稍后重试
500	DMS.00501002	Failed to query agency roleId.	根据传入的角色名称查询不到对应的角色id	请检查传入的角色名称是否正确
500	DMS.00501003	Failed to grant role to agency.	给委托授权限策略失败	请稍后重试，或联系技术支持
500	DMS.00501010	The product specification does not exist.	查询产品规格不存在。	请联系技术支持。
500	DMS.00501011	Failed to query the product ID from CBC.	包周期实例变更时向cbc查询产品id失败。	请联系技术支持。
500	DMS.00501012	Smart Connect tasks exist.	Smart Connect中存在任务。	请先删除Smart Connect中的任务。
500	DMS.10250002	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.10250003	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.10250004	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.10250005	Internal communication error.	内部通讯异常。	请联系技术支持。
500	DMS.10250006	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.10550035	tag_type must be either or or and.	tag_type不正确，tag_type必须为or或者and。	请检查tag_type

状态码	错误码	错误信息	描述	处理措施
501	DMS.111501026	Query reach maximum byte	查询消息的总字节数超过限定值	缩短时间范围，确保查询字节数不超过限定值，或者使用其他方式查询
503	DMS.111501025	Query Busy. Please try again later.	查询繁忙，请稍后查询	请稍后查询

7.3 实例状态说明

表 7-2 实例状态说明

状态	说明
CREATING	申请实例后，在实例状态进入运行中之前的状态。
RUNNING	实例正常运行状态。在这个状态的实例可以运行您的业务。
ERROR	实例处于故障的状态。
RESTARTING	实例正在进行重启操作。
STARTING	实例从已冻结到运行中的中间状态。
EXTENDING	实例正在进行规格变更操作。
EXTENDEDFAILED	实例处于规格变更操作失败的状态。
FROZEN	实例处于已冻结状态，用户可以在“我的订单”中续费开启冻结的实例。
FREEZING	实例从运行中到已冻结的中间状态。
UPGRADING	实例正在进行升级操作。
ROLLINGBACK	实例正在进行回滚操作。

7.4 获取项目 ID

操作场景

在调用接口的时候，部分URL中需要填入项目ID，所以需要获取到项目ID。有如下两种获取方式：

- **调用API获取项目ID**
- **从控制台获取项目ID**

调用 API 获取项目 ID

项目ID可以通过调用**查询指定条件下的项目信息**API获取。

获取项目ID的接口为“GET https://{Endpoint}/v3/projects”，其中{Endpoint}为IAM的终端节点，可以从**地区和终端节点**获取。接口的认证鉴权请参见**认证鉴权**。

响应示例如下，其中projects下的“id”即为项目ID。

```
{
    "projects": [
        {
            "domain_id": "65382450e8f64ac0870cd180d14e684b",
            "is_domain": false,
            "parent_id": "65382450e8f64ac0870cd180d14e684b",
            "name": "xxx-xxx-xxx",
            "description": "",
            "links": {
                "next": null,
                "previous": null,
                "self": "https://www.example.com/v3/projects/a4a5d4098fb4474fa22cd05f897d6b99"
            },
            "id": "a4a5d4098fb4474fa22cd05f897d6b99",
            "enabled": true
        }
    ],
    "links": {
        "next": null,
        "previous": null,
        "self": "https://www.example.com/v3/projects"
    }
}
```

从控制台获取项目 ID

在调用接口的时候，部分URL中需要填入项目ID（ project_id ），所以需要先在管理控制台上获取到项目ID。

项目ID获取步骤如下：

步骤1 登录管理控制台。

步骤2 鼠标悬停在右上角的用户名，选择下拉列表中的“我的凭证”。

在“API凭证”页面的项目列表中查看项目ID。

图 7-1 查看项目 ID

----结束

7.5 获取账号名和账号 ID

在调用接口的时候，部分URL中需要填入账号名和账号ID，所以需要先在管理控制台上获取到账号名和账号ID。账号名和账号ID获取步骤如下：

1. 登录管理控制台。
2. 鼠标悬停在右上角的用户名，选择下拉列表中的“我的凭证”。
   查看账号名和账号ID。

图 7-2 查看账号名和账号 ID

A 修订记录

发布日期	修订记录
2023-12-15	本次变更如下： ● 新增**查询实例的扩容规格列表**和**实例规格变更**接口。
2023-06-06	本次变更如下： ● 新增**新建元数据迁移任务**、**查询实例下所有迁移任务或查询指定迁移任务信息**和**删除元数据迁移任务**接口。
2023-04-21	本次变更如下： ● 新增**查询消费者列表**接口。
2023-02-17	本次变更如下： ● 更新**创建实例**的URI。 ● 新增**批量添加或删除实例标签**、**查询实例标签**和**查询项目标签**接口。
2023-01-13	本次变更如下： ● 新增**查询主题列表**接口。
2022-06-30	第一次正式发布。