分布式消息服务 Kafka 版

API 参考

文档版本 14

发布日期 2025-03-05

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

欢迎使用分布式消息服务Kafka版。分布式消息服务Kafka版是一款基于开源社区版Kafka提供的消息队列服务，向用户提供计算、存储和带宽资源独占式的Kafka专享实例。使用分布式消息服务Kafka版，资源按需申请，按需配置Topic的分区与副本数量，即买即用，您将有更多精力专注于业务快速开发，不用考虑部署和运维。

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

分布式消息服务Kafka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**。

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。

基本概念

- 账号

  用户注册时的账号，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（Region）

  从地理位置和网络时延维度划分，同一个Region内共享弹性计算、块存储、对象存储、VPC网络、弹性公网IP、镜像等公共服务。Region分为通用Region和专属Region，通用Region指面向公共租户提供通用云服务的Region；专属Region指只承载同一类业务或只面向特定租户提供业务服务的专用Region。

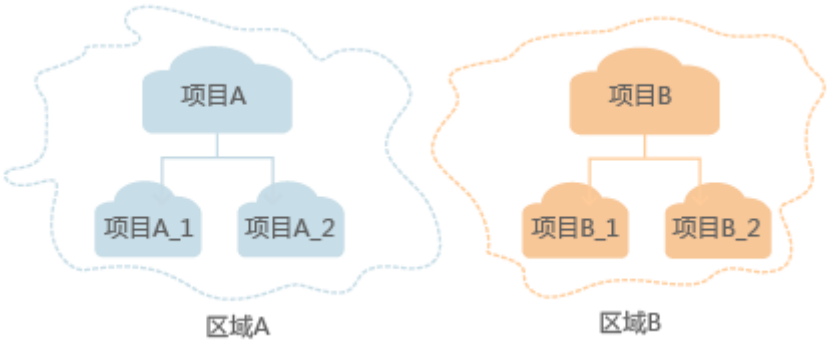
- 可用区（AZ，Availability Zone）

  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 ● 查询所有实例列表 ● 查询指定实例 ● 删除指定的实例 ● 修改实例信息 ● 批量重启或删除实例 ● 获取实例配置 ● 修改实例配置 ● 实例升级

API	说明
实例管理	包括： • 重置密码 • 重置Manager密码 • 重启Manager • 开启或关闭实例自动创建Topic功能 • 修改实例跨VPC访问的内网IP • 查询Kafka集群元数据信息 • 查询Kafka实例的协调器信息 • 修改Kafka实例Topic分区的副本 • 修改Kafka的接入方式 • 查询Topic的磁盘存储情况 • Kafka实例开始分区平衡任务 • 关闭Kafka Manager • 删除用户/客户端流控配置 • 查询用户/客户端流控配置 • 创建用户/客户端流控配置 • 修改用户/客户端流控配置
Smart Connect	包括： • 开启Smart Connect（按需实例） • 关闭Smart Connect（按需实例） • 创建Smart Connect任务 • 查询Smart Connect任务列表 • 查询Smart Connect任务详情 • 删除Smart Connect任务 • 暂停Smart Connect任务 • 启动已暂停的Smart Connect任务 • 启动未启动的Smart Connect任务/重启已暂停或者运行中的Smart Connect任务
规格变更管理	包括： • 实例扩容 • 查询实例的扩容规格列表

API	说明
主题管理	包括： • Kafka生产消息 • Kafka实例创建Topic • Kafka实例查询Topic • 修改Kafka实例Topic • Kafka实例批量删除Topic • 查询Topic的分区列表 • 查询Topic的当前生产者列表 • 查询Kafka实例Topic详细信息
消费组管理	包括： • 查询消费组信息 • 查询所有消费组 • Kafka实例批量删除消费组 • 创建消费组 • 重置消费组消费进度到指定位置 • 查询消费组消息位点 • 修改所有消费组 • 查询指定消费组 • 删除指定消费组 • 修改指定消费组 • 查询指定消费组的topic • 查询指定消费组的消费成员
用户管理	包括： • 查询用户列表 • 创建用户 • 批量删除用户 • 重置用户密码 • 修改用户参数 • 查询用户权限 • 设置用户权限

API	说明
消息管理	包括： • 查询消息 • 查询分区指定偏移量的消息 • 查询分区指定时间段的消息 • 查询分区最早消息的位置 • 查询分区最新消息的位置 • Kafka删除消息
后台任务管理	包括： • 查询实例的后台任务列表 • 查询后台任务管理中的指定记录 • 删除后台任务管理中的指定记录
标签管理	包括： • 批量添加或删除实例标签 • 查询实例标签 • 查询项目标签
诊断管理	包括： • 消息积压诊断预检查 • 创建消息积压诊断任务 • 查询消息积压诊断报告列表 • 批量删除消息积压诊断报告 • 查询诊断报告详情
其他接口	包括： • 查询维护时间窗时间段 • 查询可用区信息 • 查询产品规格列表 • 查询实例在CES的监控层级关系 • 查询Kafka产品规格核数

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**管理员创建IAM用户**来说明如何调用API。

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“管理员创建IAM用户”API的resource-path为“/v3.0/OS-USER/users”。
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。

例如您需要创建IAM用户，由于IAM为全局服务，则使用任一区域的Endpoint（比如“中国-香港”区域的Endpoint：“iam.ap-southeast-1.myhuaweicloud.com”），

并在**管理员创建IAM用户**的URI部分找到resource-path（/v3.0/OS-USER/users），拼接起来如下所示。

```
https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
```

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**管理员创建IAM用户**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/v3.0/OS-USER/users
```

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
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**管理员创建IAM用户**接口，使用AK/SK方式认证时，添加消息头后的请求如下所示。

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date,
Signature=**********
```

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**管理员创建IAM用户**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，其中加粗的字段需要根据实际值填写。

- **accountid**为IAM用户所属的账号ID。
- **username**为要创建的IAM用户名。
- **email**为IAM用户的邮箱。
- **************为IAM用户的登录密码。

```
POST https://iam.ap-southeast-1.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Sdk-Date: 20240416T095341Z
Authorization: SDK-HMAC-SHA256 Access=**********, SignedHeaders=content-type;host;x-sdk-date,
Signature=**********

{
    "user": {
        "domain_id": "accountid",
        "name": "username",
        "password": "**********",
        "email": "email",
        "description": "IAM User Description"
    }
}
```

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。

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。

- 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：通过Token认证调用请求。

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。

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",    //IAM用户名
                    "password": $ADMIN_PASS    //IAM用户密码，建议在配置文件或者环境变量中密文存放，使用时解密，确保安全
                    "domain": {
                        "name": "domainname"    //IAM用户所属账号名
                    }
                }
            }
        },
        "scope": {
            "project": {
                "name": "xxxxxxxx"    //项目名称
            }
        }
    }
}
```

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/v3.0/OS-USER/users
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

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**管理员创建IAM用户**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**管理员创建IAM用户**接口，返回如**图3-2**所示的消息头。

图 3-2 管理员创建 IAM 用户响应消息头

```
"X-Frame-Options": "SAMEORIGIN",
"X-IAM-ETag-id": "2562365939-d8f6f12921974cb097338ac11fceac8a",
"Transfer-Encoding": "chunked",
"Strict-Transport-Security": "max-age=31536000; includeSubdomains;",
"Server": "api-gateway",
"X-Request-Id": "af2953f2bcc67a42325a69a19e6c32a2",
"X-Content-Type-Options": "nosniff",
"Connection": "keep-alive",
"X-Download-Options": "noopen",
"X-XSS-Protection": "1; mode=block;",
"X-IAM-Trace-Id": "token_[redacted]_null_af2953f2bcc67a42325a69a19e6c32a2",
"Date": "Tue, 21 May 2024 09:03:40 GMT",
"Content-Type": "application/json; charset=utf8"
```

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**管理员创建IAM用户**接口，返回如下消息体。为篇幅起见，这里只展示部分内容。

```
{
    "user": {
        "id": "c131886aec...",
        "name": "IAMUser",
        "description": "IAM User Description",
        "areacode": "",
        "phone": "",
        "email": "***@***.com",
        "status": null,
```

```
        "enabled": true,
        "pwd_status": false,
        "access_mode": "default",
        "is_domain_owner": false,
        "xuser_id": "",
        "xuser_type": "",
        "password_expires_at": null,
        "create_time": "2024-05-21T09:03:41.000000",
        "domain_id": "d78cbac1.........",
        "xdomain_id": "30086000........",
        "xdomain_type": "",
        "default_project_id": null
    }
}
```

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Kafka实例。

API调用方法请参考**如何调用API**。

前提条件

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

创建 Kafka 实例

如下示例是创建Kafka实例的请求消息：

```
{
 "name": "kafka-demo",
 "engine": "kafka",
 "engine_version": 2.7,
 "broker_num": 3,
 "storage_space": 300,
 "vpc_id": "ead6c5ff-xxx-9ba91820e72c",
 "security_group_id": "aa75ae22-xxx-a9dec8c73220",
 "subnet_id": "3cb6afa2-xxx-05a7f671d6a8",
 "available_zones": [
  "effdcbcxxx6b42f56533"
 ],
 "product_id": "c6.2u4g.cluster",
 "storage_spec_code": "dms.physical.storage.high.v2"
}
```

- name：实例名称，由您自行定义。
- engine：消息引擎，设置kafka。
- engine_version：消息引擎的版本。
- broker_num：代理个数。
- storage_space：消息存储空间，单位GB。具体取值范围，请参考**创建实例**。
- vpc_id：Kafka实例所在的VPC（虚拟私有云）的ID。请参考**创建实例**获取。
- security_group_id：安全组ID。请参考**创建实例**获取。
- subnet_id：VPC内子网的网络ID。请参考**创建实例**获取。

- 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：产品标识。请参考**查询产品规格列表**获取。
- storage_spec_code：存储IO规格。具体取值范围，请参考**创建实例**。

5 API V2（推荐）

5.1 生命周期管理

5.1.1 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 5-1 路径参数

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

请求参数

表 5-2 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：kafka。
engine_version	是	String	消息引擎的版本。取值填写为： ● 1.1.0 ● 2.7 ● 3.x
broker_num	是	Integer	代理个数。
storage_space	是	Integer	消息存储空间，单位GB。 ● Kafka实例规格为c6.2u4g.cluster时，存储空间取值范围300GB ~ 300000GB。 ● Kafka实例规格为c6.4u8g.cluster时，存储空间取值范围300GB ~ 600000GB。 ● Kafka实例规格为c6.8u16g.cluster时，存储空间取值范围300GB ~ 1500000GB。 ● Kafka实例规格为c6.12u24g.cluster时，存储空间取值范围300GB ~ 1500000GB。 ● Kafka实例规格为c6.16u32g.cluster时，存储空间取值范围300GB ~ 1500000GB。

参数	是否必选	参数类型	描述
access_user	否	String	当ssl_enable为true时，该参数必选，ssl_enable为false时，该参数无效。 认证用户名，只能由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4~64的字符。
password	否	String	当ssl_enable为true时，该参数必选，ssl_enable为false时，该参数无效。 实例的认证密码。 复杂度要求： ● 输入长度为8到32位的字符串。 ● 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（`~!@#$%^&*()-_=+|[{}]:'",<.>/?）和空格，并且不能以-开头
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。
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。
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。

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。 创建Kafka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	产品ID。 产品ID可以从**查询产品规格列表**获取。
maintain_begin	否	String	维护时间窗开始时间，格式为HH:mm。
maintain_end	否	String	维护时间窗结束时间，格式为HH:mm。
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
tenant_ips	否	Array of strings	创建实例时可以手动指定实例节点的内网IP地址，仅支持指定IPv4地址。 指定内网IP地址数量必须小于等于创建的节点数量。 如果指定的内网IP地址数量小于创建的节点数量时，系统会自动为剩余的节点随机分配内网IP地址。
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。
ssl_enable	否	Boolean	是否开启SASL加密访问。 • true：开启SASL加密访问。 • false：关闭SASL加密访问。

参数	是否必选	参数类型	描述
kafka_security_protocol	否	String	开启SASL后使用的安全协议。 ● SASL_SSL: 使用SSL证书加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT: 通过明文传输，支持账号密码认证，性能更好。 若该字段值为空，默认开启SASL_SSL认证机制。实例创建后，此参数不支持动态修改。 若创建实例时，使用了port_protocol参数，则Kafka的内网访问安全协议以及公网访问安全协议会使用port_protocol中的值，则此参数无效。
sasl_enabled_mechanisms	否	Array of strings	开启SASL后使用的认证机制，如果开启了SASL认证功能（即ssl_enable=true），该字段为必选。 若该字段值为空，默认开启PLAIN认证机制。 选择其一进行SASL认证即可，支持同时开启两种认证机制。 取值如下： ● PLAIN: 简单的用户名密码校验。 ● SCRAM-SHA-512: 用户凭证校验，安全性比PLAIN机制更高。
port_protocol	否	**PortProtocol** object	设置Kafka实例的接入方式。PLAINTEXT表示明文接入，SASL_SSL或者SASL_PLAINTEEXT表示密文接入。 内网访问不支持关闭，明文接入和密文接入至少开启一个。 跨VPC访问的安全协议等于内网访问的安全协议，若内网同时开启了密文访问和明文访问，则跨VPC访问的安全协议会优先使用密文访问的安全协议。

参数	是否必选	参数类型	描述
retention_policy	否	String	磁盘的容量到达容量阈值后，对于消息的处理策略。 取值如下： • produce_reject：表示拒绝消息写入。 • time_base：表示自动删除最老消息。
ipv6_enable	否	Boolean	是否开启ipv6。仅在虚拟私有云支持ipv6时生效。
disk_encrypted_enable	否	Boolean	是否开启磁盘加密。
disk_encrypted_key	否	String	磁盘加密key，未开启磁盘加密时为空
connector_enable	否	Boolean	是否开启消息转储功能。 默认不开启消息转储。
enable_auto_topic	否	Boolean	是否打开kafka自动创建topic功能。 • true：开启 • false：关闭 当您选择开启，表示生产或消费一个未创建的Topic时，会自动创建一个包含3个分区和3个副本的Topic。 默认是false关闭。
storage_spec_code	是	String	存储IO规格。 取值范围： • dms.physical.storage.high.v2：使用高IO的磁盘类型。 • dms.physical.storage.ultra.v2：使用超高IO的磁盘类型。 如何选择磁盘类型请参考《 云硬盘 产品介绍 》的“磁盘类型及性能介绍”。
enterprise_project_id	否	String	企业项目ID。若为企业项目账号，该参数必填。
tags	否	Array of **TagEntity** objects	标签列表。

参数	是否必选	参数类型	描述
arch_type	否	String	CPU架构。当前只支持X86架构。 取值范围： ● X86
vpc_client_plain	否	Boolean	VPC内网明文访问。
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 5-3 PortProtocol

参数	是否必选	参数类型	描述
private_plain_enable	否	Boolean	是否开启内网明文访问连接方式。 取值范围： ● true: 开启内网明文访问连接方式，连接地址：ip:9092，访问协议PLAINTEXT。 ● false: 关闭内网明文访问。 默认为false。
private_sasl_ssl_enable	否	Boolean	是否开启安全协议为SASL_SSL的内网密文接入方式。 取值范围： ● true: 开启安全协议为SASL_SSL的内网密文接入方式。 private_sasl_ssl_enable和private_sasl_plaintext_enable不能同时为true。 ● false: 关闭安全协议为SASL_SSL的内网接入方式。 默认为false。

参数	是否必选	参数类型	描述
private_sasl_plaintext_enable	否	Boolean	是否开启安全协议为SASL_PLAINTEXT的内网密文接入方式。 取值范围： ● true: 开启安全协议为SASL_PLAINTEXT的内网密文接入方式，连接地址：ip:9093，访问协议SASL_PLAINTEXT。 private_sasl_plaintext_enable和private_sasl_ssl_enable不能同时为true。 ● false: 关闭安全协议为SASL_PLAINTEXT的内网密文接入方式。 默认为false。
public_plain_enable	否	Boolean	是否开启公网明文访问连接方式。 取值范围： ● true: 开启公网明文访问连接方式，连接地址：ip:9094，访问协议PLAINTEXT。 开启公网明文接入前，需要先开启公网访问功能。 ● false: 关闭公网明文接入方式。 默认为false。
public_sasl_ssl_enable	否	Boolean	是否开启安全协议为SASL_SSL的公网密文接入。 取值范围： ● true: 开启安全协议为SASL_SSL的公网密文接入方式，连接地址：ip:9095，访问协议：SASL_SSL。 public_sasl_ssl_enable和public_sasl_plaintext_enable不能同时为true。 为true时，需要实例开启公网。 ● false: 关闭安全协议为SASL_SSL的公网密文接入方式。 默认为false。

参数	是否必选	参数类型	描述
public_sasl_plaintext_enable	否	Boolean	是否开启安全协议为SASL_PLAINTEXT的公网密文接入方式。 取值范围： ● true: 开启安全协议为SASL_PLAINTEXT的公网密文接入方式，连接地址：ip:9095，访问协议：SASL_PLAINTEXT。 public_sasl_plaintext_enable和public_sasl_ssl_enable不能同时为true。 为true时，需要实例开启公网。 ● false: 关闭安全协议为SASL_PLAINTEXT的公网密文接入方式。 默认为false。

表 5-4 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 _ . : = + - @ ● 不能以_sys_开头。 ● 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 _ . : = + - @

表 5-5 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。
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 5-6 响应 Body 参数

参数	参数类型	描述
instance_id	String	实例ID

请求示例

- 创建一个按需付费的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

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

{
  "name" : "kafka-test",
  "description" : "",
  "engine" : "kafka",
  "engine_version" : "2.7",
  "storage_space" : 300,
  "vpc_id" : "********-9b4a-44c5-a964-**********",
  "subnet_id" : "********-8fbf-4438-ba71-**********",
  "security_group_id" : "********-e073-4aad-991f-**********",
  "available_zones" : [ "********706d4c1fb0eb72f0********" ],
  "product_id" : "c6.2u4g.cluster",
  "ssl_enable" : true,
  "kafka_security_protocol" : "SASL_SSL",
  "sasl_enabled_mechanisms" : [ "SCRAM-SHA-512" ],
  "storage_spec_code" : "dms.physical.storage.ultra.v2",
  "broker_num" : 3,
  "arch_type" : "X86",
  "enterprise_project_id" : "0",
  "access_user" : "********",
  "password" : "********",
  "enable_publicip" : true,
  "tags" : [ {
    "key" : "aaa",
    "value" : "111"
  } ],
  "retention_policy" : "time_base",
  "disk_encrypted_enable" : true,
  "disk_encrypted_key" : "********-b953-4875-a743-**********",
  "publicip_id" : "********-88fc-4a8c-86d0-**********,********-16af-455d-8d54-
**********,********-3d69-4367-95ab-**********",
  "vpc_client_plain" : true,
  "enable_auto_topic" : true,
  "tenant_ips" : [ "127.xx.xx.x", "127.xx.xx.x", "127.xx.xx.x" ]
}
```

- 创建一个包年包月的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

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

{
  "name" : "kafka-test1",
  "description" : "",
  "engine" : "kafka",
  "engine_version" : "2.7",
  "storage_space" : 300,
  "vpc_id" : "********-9b4a-44c5-a964-**********",
  "subnet_id" : "********-8fbf-4438-ba71-**********",
```

```
  "security_group_id" : "********-e073-4aad-991f-************",
  "available_zones" : [ "********706d4c1fb0eb72f0********" ],
  "product_id" : "c6.2u4g.cluster",
  "ssl_enable" : true,
  "kafka_security_protocol" : "SASL_SSL",
  "sasl_enabled_mechanisms" : [ "SCRAM-SHA-512" ],
  "storage_spec_code" : "dms.physical.storage.ultra.v2",
  "broker_num" : 3,
  "arch_type" : "X86",
  "enterprise_project_id" : "0",
  "access_user" : "********",
  "password" : "********",
  "enable_publicip" : true,
  "tags" : [ {
    "key" : "aaa",
    "value" : "111"
  } ],
  "retention_policy" : "time_base",
  "publicip_id" : "********-88fc-4a8c-86d0-************,********-16af-455d-8d54-
************,********-3d69-4367-95ab-************",
  "vpc_client_plain" : true,
  "enable_auto_topic" : true,
  "bss_param" : {
    "charging_mode" : "prePaid",
    "period_type" : "month",
    "period_num" : 1,
    "is_auto_pay" : true
  },
  "tenant_ips" : [ "127.xx.xx.x", "127.xx.xx.x", "127.xx.xx.x" ]
}
```

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

- 创建一个按需付费的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateInstanceByEngineRequest request = new CreateInstanceByEngineRequest();
        request.withEngine(CreateInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
        CreateInstanceByEngineReq body = new CreateInstanceByEngineReq();
        List<TagEntity> listbodyTags = new ArrayList<>();
        listbodyTags.add(
            new TagEntity()
                .withKey("aaa")
                .withValue("111")
        );
        List<CreateInstanceByEngineReq.SaslEnabledMechanismsEnum> listbodySaslEnabledMechanisms = new ArrayList<>();

listbodySaslEnabledMechanisms.add(CreateInstanceByEngineReq.SaslEnabledMechanismsEnum.fromValue("SCRAM-SHA-512"));
        List<String> listbodyAvailableZones = new ArrayList<>();
        listbodyAvailableZones.add("********706d4c1fb0eb72f0********");
        body.withVpcClientPlain(true);
        body.withArchType("X86");
        body.withTags(listbodyTags);
        body.withEnterpriseProjectId("0");

body.withStorageSpecCode(CreateInstanceByEngineReq.StorageSpecCodeEnum.fromValue("dms.physical.storage.ultra.v2"));
        body.withEnableAutoTopic(true);
        body.withDiskEncryptedKey("********-b953-4875-a743-************");
        body.withDiskEncryptedEnable(true);

body.withRetentionPolicy(CreateInstanceByEngineReq.RetentionPolicyEnum.fromValue("time_base"));
        body.withSaslEnabledMechanisms(listbodySaslEnabledMechanisms);
        body.withKafkaSecurityProtocol("SASL_SSL");
        body.withSslEnable(true);
        body.withPublicipId("********-88fc-4a8c-86d0-************,********-16af-455d-8d54-************,********-3d69-4367-95ab-************");
        body.withEnablePublicip(true);
        body.withProductId("c6.2u4g.cluster");
        body.withAvailableZones(listbodyAvailableZones);
        body.withSubnetId("********-8fbf-4438-ba71-************");
        body.withSecurityGroupId("********-e073-4aad-991f-************");
        body.withVpcId("********-9b4a-44c5-a964-************");
        body.withPassword("********");
        body.withAccessUser("********");
        body.withStorageSpace(300);
        body.withBrokerNum(3);
        body.withEngineVersion("2.7");
        body.withEngine(CreateInstanceByEngineReq.EngineEnum.fromValue("kafka"));
        body.withDescription("");
        body.withName("kafka-test");
        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());
        }
    }
}
```

- 创建一个包年包月的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateInstanceByEngineRequest request = new CreateInstanceByEngineRequest();
        request.withEngine(CreateInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
        CreateInstanceByEngineReq body = new CreateInstanceByEngineReq();
        BssParam bssParambody = new BssParam();
        bssParambody.withChargingMode(BssParam.ChargingModeEnum.fromValue("prePaid"))
            .withIsAutoPay(true)
            .withPeriodType(BssParam.PeriodTypeEnum.fromValue("month"))
            .withPeriodNum(1);
        List<TagEntity> listbodyTags = new ArrayList<>();
        listbodyTags.add(
            new TagEntity()
                .withKey("aaa")
                .withValue("111")
        );
        List<CreateInstanceByEngineReq.SaslEnabledMechanismsEnum> listbodySaslEnabledMechanisms = new ArrayList<>();
```

```
listbodySaslEnabledMechanisms.add(CreateInstanceByEngineReq.SaslEnabledMechanismsEnum.fromV
alue("SCRAM-SHA-512"));
        List<String> listbodyAvailableZones = new ArrayList<>();
        listbodyAvailableZones.add("********706d4c1fb0eb72f0********");
        body.withBssParam(bssParambody);
        body.withVpcClientPlain(true);
        body.withArchType("X86");
        body.withTags(listbodyTags);
        body.withEnterpriseProjectId("0");

body.withStorageSpecCode(CreateInstanceByEngineReq.StorageSpecCodeEnum.fromValue("dms.physic
al.storage.ultra.v2"));
        body.withEnableAutoTopic(true);

body.withRetentionPolicy(CreateInstanceByEngineReq.RetentionPolicyEnum.fromValue("time_base"));
        body.withSaslEnabledMechanisms(listbodySaslEnabledMechanisms);
        body.withKafkaSecurityProtocol("SASL_SSL");
        body.withSslEnable(true);
        body.withPublicipId("********-88fc-4a8c-86d0-************,********-16af-455d-8d54-
************,********-3d69-4367-95ab-************");
        body.withEnablePublicip(true);
        body.withProductId("c6.2u4g.cluster");
        body.withAvailableZones(listbodyAvailableZones);
        body.withSubnetId("********-8fbf-4438-ba71-************");
        body.withSecurityGroupId("********-e073-4aad-991f-************");
        body.withVpcId("********-9b4a-44c5-a964-************");
        body.withPassword("********");
        body.withAccessUser("********");
        body.withStorageSpace(300);
        body.withBrokerNum(3);
        body.withEngineVersion("2.7");
        body.withEngine(CreateInstanceByEngineReq.EngineEnum.fromValue("kafka"));
        body.withDescription("");
        body.withName("kafka-test1");
        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

- 创建一个按需付费的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateInstanceByEngineRequest()
        request.engine = "{engine}"
        listTagsbody = [
            TagEntity(
                key="aaa",
                value="111"
            )
        ]
        listSaslEnabledMechanismsbody = [
            "SCRAM-SHA-512"
        ]
        listAvailableZonesbody = [
            "********706d4c1fb0eb72f0********"
        ]
        request.body = CreateInstanceByEngineReq(
            vpc_client_plain=True,
            arch_type="X86",
            tags=listTagsbody,
            enterprise_project_id="0",
            storage_spec_code="dms.physical.storage.ultra.v2",
            enable_auto_topic=True,
            disk_encrypted_key="********-b953-4875-a743-************",
            disk_encrypted_enable=True,
            retention_policy="time_base",
            sasl_enabled_mechanisms=listSaslEnabledMechanismsbody,
            kafka_security_protocol="SASL_SSL",
            ssl_enable=True,
            publicip_id="********-88fc-4a8c-86d0-************,********-16af-455d-8d54-
************,********-3d69-4367-95ab-************",
            enable_publicip=True,
            product_id="c6.2u4g.cluster",
            available_zones=listAvailableZonesbody,
            subnet_id="********-8fbf-4438-ba71-************",
            security_group_id="********-e073-4aad-991f-************",
            vpc_id="********-9b4a-44c5-a964-************",
            password="********",
            access_user="********",
            storage_space=300,
            broker_num=3,
            engine_version="2.7",
            engine="kafka",
            description="",
            name="kafka-test"
        )
        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)
```

- 创建一个包年包月的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateInstanceByEngineRequest()
        request.engine = "{engine}"
        bssParambody = BssParam(
            charging_mode="prePaid",
            is_auto_pay=True,
            period_type="month",
            period_num=1
        )
        listTagsbody = [
            TagEntity(
                key="aaa",
                value="111"
            )
        ]
        listSaslEnabledMechanismsbody = [
            "SCRAM-SHA-512"
        ]
        listAvailableZonesbody = [
            "********706d4c1fb0eb72f0********"
        ]
        request.body = CreateInstanceByEngineReq(
            bss_param=bssParambody,
            vpc_client_plain=True,
            arch_type="X86",
            tags=listTagsbody,
            enterprise_project_id="0",
            storage_spec_code="dms.physical.storage.ultra.v2",
            enable_auto_topic=True,
            retention_policy="time_base",
            sasl_enabled_mechanisms=listSaslEnabledMechanismsbody,
            kafka_security_protocol="SASL_SSL",
            ssl_enable=True,
            publicip_id="********-88fc-4a8c-86d0-************,********-16af-455d-8d54-************,********-3d69-4367-95ab-************",
            enable_publicip=True,
            product_id="c6.2u4g.cluster",
            available_zones=listAvailableZonesbody,
            subnet_id="********-8fbf-4438-ba71-************",
            security_group_id="********-e073-4aad-991f-************",
            vpc_id="********-9b4a-44c5-a964-************",
            password="********",
            access_user="********",
```

```
            storage_space=300,
            broker_num=3,
            engine_version="2.7",
            engine="kafka",
            description="",
            name="kafka-test1"
        )
        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

- 创建一个按需付费的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateInstanceByEngineRequest{}
    request.Engine = model.GetCreateInstanceByEngineRequestEngineEnum().ENGINE
    keyTags:= "aaa"
    valueTags:= "111"
    var listTagsbody = []model.TagEntity{
        {
            Key: &keyTags,
            Value: &valueTags,
        },
    }
    var listSaslEnabledMechanismsbody = []model.CreateInstanceByEngineReqSaslEnabledMechanisms{
        model.GetCreateInstanceByEngineReqSaslEnabledMechanismsEnum().SCRAM_SHA_512,
    }
    var listAvailableZonesbody = []string{
        "********706d4c1fb0eb72f0********",
    }
```

```
    vpcClientPlainCreateInstanceByEngineReq:= true
    archTypeCreateInstanceByEngineReq:= "X86"
    enterpriseProjectIdCreateInstanceByEngineReq:= "0"
    enableAutoTopicCreateInstanceByEngineReq:= true
    diskEncryptedKeyCreateInstanceByEngineReq:= "********-b953-4875-a743-************"
    diskEncryptedEnableCreateInstanceByEngineReq:= true
    retentionPolicyCreateInstanceByEngineReq:=
model.GetCreateInstanceByEngineReqRetentionPolicyEnum().TIME_BASE
    kafkaSecurityProtocolCreateInstanceByEngineReq:= "SASL_SSL"
    sslEnableCreateInstanceByEngineReq:= true
    publicipIdCreateInstanceByEngineReq:= "********-88fc-4a8c-86d0-************,********-16af-455d-8d54-
************,********-3d69-4367-95ab-************"
    enablePublicipCreateInstanceByEngineReq:= true
    passwordCreateInstanceByEngineReq:= "********"
    accessUserCreateInstanceByEngineReq:= "********"
    descriptionCreateInstanceByEngineReq:= ""
    request.Body = &model.CreateInstanceByEngineReq{
        VpcClientPlain: &vpcClientPlainCreateInstanceByEngineReq,
        ArchType: &archTypeCreateInstanceByEngineReq,
        Tags: &listTagsbody,
        EnterpriseProjectId: &enterpriseProjectIdCreateInstanceByEngineReq,
        StorageSpecCode:
model.GetCreateInstanceByEngineReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_ULTRA,
        EnableAutoTopic: &enableAutoTopicCreateInstanceByEngineReq,
        DiskEncryptedKey: &diskEncryptedKeyCreateInstanceByEngineReq,
        DiskEncryptedEnable: &diskEncryptedEnableCreateInstanceByEngineReq,
        RetentionPolicy: &retentionPolicyCreateInstanceByEngineReq,
        SaslEnabledMechanisms: &listSaslEnabledMechanismsbody,
        KafkaSecurityProtocol: &kafkaSecurityProtocolCreateInstanceByEngineReq,
        SslEnable: &sslEnableCreateInstanceByEngineReq,
        PublicipId: &publicipIdCreateInstanceByEngineReq,
        EnablePublicip: &enablePublicipCreateInstanceByEngineReq,
        ProductId: "c6.2u4g.cluster",
        AvailableZones: listAvailableZonesbody,
        SubnetId: "********-8fbf-4438-ba71-************",
        SecurityGroupId: "********-e073-4aad-991f-************",
        VpcId: "********-9b4a-44c5-a964-************",
        Password: &passwordCreateInstanceByEngineReq,
        AccessUser: &accessUserCreateInstanceByEngineReq,
        StorageSpace: int32(300),
        BrokerNum: int32(3),
        EngineVersion: "2.7",
        Engine: model.GetCreateInstanceByEngineReqEngineEnum().KAFKA,
        Description: &descriptionCreateInstanceByEngineReq,
        Name: "kafka-test",
    }
    response, err := client.CreateInstanceByEngine(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建一个包年包月的Kafka实例，版本为2.7，规格为2U4G*3，300GB的存储空间。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateInstanceByEngineRequest{}
    request.Engine = model.GetCreateInstanceByEngineRequestEngineEnum().ENGINE
    chargingModeBssParam:= model.GetBssParamChargingModeEnum().PRE_PAID
    isAutoPayBssParam:= true
    periodTypeBssParam:= model.GetBssParamPeriodTypeEnum().MONTH
    periodNumBssParam:= int32(1)
    bssParambody := &model.BssParam{
        ChargingMode: &chargingModeBssParam,
        IsAutoPay: &isAutoPayBssParam,
        PeriodType: &periodTypeBssParam,
        PeriodNum: &periodNumBssParam,
    }
    keyTags:= "aaa"
    valueTags:= "111"
    var listTagsbody = []model.TagEntity{
        {
            Key: &keyTags,
            Value: &valueTags,
        },
    }
    var listSaslEnabledMechanismsbody = []model.CreateInstanceByEngineReqSaslEnabledMechanisms{
        model.GetCreateInstanceByEngineReqSaslEnabledMechanismsEnum().SCRAM_SHA_512,
    }
    var listAvailableZonesbody = []string{
        "********706d4c1fb0eb72f0********",
    }
    vpcClientPlainCreateInstanceByEngineReq:= true
    archTypeCreateInstanceByEngineReq:= "X86"
    enterpriseProjectIdCreateInstanceByEngineReq:= "0"
    enableAutoTopicCreateInstanceByEngineReq:= true
    retentionPolicyCreateInstanceByEngineReq:=
model.GetCreateInstanceByEngineReqRetentionPolicyEnum().TIME_BASE
    kafkaSecurityProtocolCreateInstanceByEngineReq:= "SASL_SSL"
    sslEnableCreateInstanceByEngineReq:= true
    publicipIdCreateInstanceByEngineReq:= "********-88fc-4a8c-86d0-************,********-16af-455d-8d54-
************,********-3d69-4367-95ab-************"
    enablePublicipCreateInstanceByEngineReq:= true
    passwordCreateInstanceByEngineReq:= "********"
    accessUserCreateInstanceByEngineReq:= "********"
    descriptionCreateInstanceByEngineReq:= ""
    request.Body = &model.CreateInstanceByEngineReq{
        BssParam: bssParambody,
        VpcClientPlain: &vpcClientPlainCreateInstanceByEngineReq,
        ArchType: &archTypeCreateInstanceByEngineReq,
        Tags: &listTagsbody,
        EnterpriseProjectId: &enterpriseProjectIdCreateInstanceByEngineReq,
        StorageSpecCode:
model.GetCreateInstanceByEngineReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_ULTRA,
        EnableAutoTopic: &enableAutoTopicCreateInstanceByEngineReq,
```

```
        RetentionPolicy: &retentionPolicyCreateInstanceByEngineReq,
        SaslEnabledMechanisms: &listSaslEnabledMechanismsbody,
        KafkaSecurityProtocol: &kafkaSecurityProtocolCreateInstanceByEngineReq,
        SslEnable: &sslEnableCreateInstanceByEngineReq,
        PublicipId: &publicipIdCreateInstanceByEngineReq,
        EnablePublicip: &enablePublicipCreateInstanceByEngineReq,
        ProductId: "c6.2u4g.cluster",
        AvailableZones: listAvailableZonesbody,
        SubnetId: "********-8fbf-4438-ba71-**********",
        SecurityGroupId: "********-e073-4aad-991f-**********",
        VpcId: "********-9b4a-44c5-a964-**********",
        Password: &passwordCreateInstanceByEngineReq,
        AccessUser: &accessUserCreateInstanceByEngineReq,
        StorageSpace: int32(300),
        BrokerNum: int32(3),
        EngineVersion: "2.7",
        Engine: model.GetCreateInstanceByEngineReqEngineEnum().KAFKA,
        Description: &descriptionCreateInstanceByEngineReq,
        Name: "kafka-test1",
    }
    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**。

5.1.2 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 5-7 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 5-8 Query 参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎：kafka。
name	否	String	实例名称。
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”，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。
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	当次查询返回的最大实例个数，默认值为10，取值范围为1~50。

请求参数

无

响应参数

状态码：200

表 5-9 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 5-10 ShowInstanceResp

参数	参数类型	描述
name	String	实例名称。
engine	String	引擎。
engine_version	String	版本。
description	String	实例描述。
specification	String	实例规格。
storage_space	Integer	消息存储空间，单位：GB。
partition_num	String	Kafka实例的分区数量。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
dns_enable	Boolean	实例是否开启域名访问功能。 • true：开启 • false：未开启
connect_address	String	实例连接IP地址。
port	Integer	实例连接端口。
status	String	实例的状态。详细状态说明请参考**实例状态说明**。
instance_id	String	实例ID。

参数	参数类型	描述
resource_spec_code	String	资源规格标识。 ● dms.instance.kafka.cluster.c3.mini：Kafka实例的基准带宽为100MByte/秒。 ● dms.instance.kafka.cluster.c3.small.2：Kafka实例的基准带宽为300MByte/秒。 ● dms.instance.kafka.cluster.c3.middle.2：Kafka实例的基准带宽为600MByte/秒。 ● dms.instance.kafka.cluster.c3.high.2：Kafka实例的基准带宽为1200MByte/秒。
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	VPC ID。
vpc_name	String	VPC的名称。
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。
subnet_name	String	子网名称。
subnet_cidr	String	子网网段。
user_id	String	用户ID。
user_name	String	用户名。
access_user	String	实例访问用户名。
order_id	String	订单ID，只有在包周期计费时才会有order_id值，其他计费方式order_id值为空。
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_publicip	Boolean	实例是否开启公网访问功能。 ● true：开启 ● false：未开启
management_connect_address	String	Kafka实例的Kafka Manager连接地址。

参数	参数类型	描述
ssl_enable	Boolean	是否开启安全认证。 ● true：开启 ● false：未开启
broker_ssl_enable	Boolean	是否开启broker间副本加密传输。 ● true：开启 ● false：未开启
kafka_security_protocol	String	Kafka使用的安全协议。 若实例详情中不存在port_protocols返回参数，则kafka_security_protocol同时代表内网访问、公网访问以及跨VPC访问的安全协议。 若实例详情中存在port_protocols返回参数，则kafka_security_protocol仅代表跨VPC访问的安全协议。内网访问公网访问的安全协议请参考port_protocols参数。 ● PLAINTEXT: 既未采用SSL证书进行加密传输，也不支持账号密码认证。性能更好，安全性较低，建议在生产环境下公网访问不使用此方式。 ● SASL_SSL: 采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT: 明文传输，支持账号密码认证，性能更好，建议使用SCRAM-SHA-512机制。
sasl_enabled_mechanisms	Array of strings	开启SASL后使用的认证机制。 ● PLAIN: 简单的用户名密码校验。 ● SCRAM-SHA-512: 用户凭证校验，安全性比PLAIN机制更高。
ssl_two_way_enable	Boolean	是否开启双向认证。
cert_replaced	Boolean	是否能够证书替换。
public_management_connect_address	String	公网访问Kafka Manager连接地址。
enterprise_project_id	String	企业项目ID。

参数	参数类型	描述
is_logical_volume	Boolean	实例扩容时用于区分老实例与新实例。 ● true：新创建的实例，允许磁盘动态扩容不需要重启。 ● false：老实例
extend_times	Integer	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。
enable_auto_topic	Boolean	是否打开kafka自动创建topic功能。 ● true：开启 ● false：关闭
type	String	实例类型：集群，cluster。
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。
available_zones	Array of strings	实例节点所在的可用区，返回“可用区ID”。
available_zone_names	Array of strings	实例节点所在的可用区名称，返回“可用区名称”。
total_storage_space	Integer	总共消息存储空间，单位：GB。
public_connect_address	String	实例公网连接IP地址。当实例开启了公网访问，实例才包含该参数。
public_connect_domain_name	String	实例公网连接域名。当实例开启了公网访问，实例才包含该参数。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	IO规格。
service_type	String	服务类型。
storage_type	String	存储类型。
retention_policy	String	消息老化策略。
kafka_public_status	String	Kafka公网开启状态。
public_bandwidth	Integer	kafka公网访问带宽。

参数	参数类型	描述
enable_log_collection	Boolean	是否开启消息收集功能。
new_auth_cert	Boolean	是否开启新证书。
cross_vpc_info	String	跨VPC访问信息。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
connector_enable	Boolean	是否开启转储。新规格产品暂不支持开启转储。
connector_node_num	Integer	connector节点数量。
connector_id	String	转储任务ID。
rest_enable	Boolean	是否开启Kafka rest功能。
rest_connect_address	String	Kafka rest连接地址。
public_boundwidth	Integer	kafka公网访问带宽。待删除版本。
message_query_inst_enable	Boolean	是否开启消息查询功能。
vpc_client_plain	Boolean	是否开启VPC明文访问。
support_features	String	Kafka实例支持的特性功能。
trace_enable	Boolean	是否开启消息轨迹功能。
agent_enable	Boolean	是否开启代理。
pod_connect_address	String	租户侧连接地址。
disk_encrypted	Boolean	是否开启磁盘加密。
disk_encrypted_key	String	磁盘加密key，未开启磁盘加密时为空。
kafka_private_connect_address	String	Kafka实例内网连接地址。
kafka_private_connect_domain_name	String	Kafka实例内网连接域名。
ces_version	String	云监控版本。

参数	参数类型	描述
public_access_enabled	String	区分实例什么时候开启的公网访问 取值范围： ● true：已开启公网访问 ● actived：已开启公网访问 ● closed：已关闭公网访问 ● false：已关闭公网访问
node_num	Integer	节点数。
port_protocols	**PortProtocolsEntity** object	实例支持的连接方式及其连接地址。
enable_acl	Boolean	是否开启访问控制。
new_spec_billing_enable	Boolean	是否启用新规格计费。
broker_num	Integer	节点数量。
tags	Array of **TagEntity** objects	标签列表。
dr_enable	Boolean	是否为容灾实例。

表 5-11 PortProtocolsEntity

参数	参数类型	描述
private_plain_enable	Boolean	实例是否支持内网PLAINTEXT访问接入方式。 ● true：实例支持内网PLAINTEXT访问方式接入方式。 ● false：实例不支持内网PLAINTEXT访问接入方式。
private_plain_address	String	kafka内网PLAINTEXT接入方式连接地址。
private_plain_domain_name	String	内网明文连接域名
private_sasl_ssl_enable	Boolean	实例是否支持内网SASL_SSL访问接入方式。 ● true：实例支持内网SASL_SSL访问方式接入方式。 ● false：实例不支持内网SASL_SSL访问接入方式。

参数	参数类型	描述
private_sasl_ssl_address	String	kafka内网SASL_SSL接入方式连接地址。
private_sasl_ssl_domain_name	String	内网SASL_SSL连接域名
private_sasl_plaintext_enable	Boolean	实例是否支持内网SASL_PLAINTEXT访问接入方式。 ● true，实例支持内网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持内网SASL_PLAINTEXT访问接入方式。
private_sasl_plaintext_address	String	kafka内网SASL_PLAINTEXT接入方式连接地址。
private_sasl_plaintext_domain_name	String	内网SASL_PLAINTEXT连接域名
public_plain_enable	Boolean	实例是否支持公网PLAINTEXT访问接入方式。 ● true，实例支持公网PLAINTEXT访问方式接入方式。 ● false，实例不支持公网PLAINTEXT访问接入方式。
public_plain_address	String	kafka公网PLAINTEXT接入方式连接地址。
public_plain_domain_name	String	公网明文连接域名
public_sasl_ssl_enable	Boolean	实例是否支持公网SASL_SSL访问接入方式。 ● true，实例支持内网SASL_SSL访问方式接入方式。 ● false，实例不支持公网SASL_SSL访问接入方式。
public_sasl_ssl_address	String	kafka公网SASL_SSL接入方式连接地址。
public_sasl_ssl_domain_name	String	公网SASL_SSL连接域名

参数	参数类型	描述
public_sasl_plaintext_enable	Boolean	实例是否支持公网SASL_PLAINTEXT访问接入方式。 ● true，实例支持公网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持公网SASL_PLAINTEXT访问接入方式。
public_sasl_plaintext_address	String	kafka公网SASL_PLAINTEXT接入方式连接地址。
public_sasl_plaintext_domain_name	String	公网SASL_PLAINTEXT连接域名

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_ . : = + - @ ● 不能以_sys_开头。 ● 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_ . : = + - @

请求示例

查询所有实例列表

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

响应示例

状态码：200

查询所有实例列表成功。

```
{
  "instances" : [ {
    "name" : "kafka-2085975099",
    "engine" : "kafka",
    "port" : 9092,
    "status" : "RUNNING",
    "type" : "cluster",
    "specification" : "100MB",
    "engine_version" : "1.1.0",
    "connect_address" : "192.168.0.100,192.168.0.61,192.168.0.72",
    "instance_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "resource_spec_code" : "dms.instance.kafka.cluster.c3.mini",
    "charging_mode" : 1,
    "vpc_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "vpc_name" : "dms-test",
    "created_at" : "1585618587087",
    "product_id" : "00300-30308-0--0",
    "security_group_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "security_group_name" : "Sys-default",
    "subnet_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "available_zones" : [ "38b0f7a602344246bcb0da47b5d548e7" ],
    "available_zone_names" : [ "AZ1" ],
    "user_id" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    "user_name" : "paas_dms",
    "maintain_begin" : "02:00:00",
    "maintain_end" : "06:00:00",
    "enable_log_collection" : false,
    "new_auth_cert" : false,
    "storage_space" : 492,
    "total_storage_space" : 600,
    "used_storage_space" : 25,
    "partition_num" : "300",
    "enable_publicip" : false,
    "ssl_enable" : false,
    "broker_ssl_enable" : false,
    "cert_replaced" : false,
    "management_connect_address" : "https://192.168.0.100:9999",
    "cross_vpc_info" : "{\"192.168.0.61\":{\"advertised_ip\":\"192.168.0.61\",\"port\":9011,\"port_id
\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.72\":{\"advertised_ip\":\"192.168.0.72\",\"port
\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.100\":{\"advertised_ip
\":\"192.168.0.100\",\"port\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"}}",
    "storage_resource_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "storage_spec_code" : "dms.physical.storage.ultra",
    "service_type" : "advanced",
    "storage_type" : "hec",
    "enterprise_project_id" : "0",
    "is_logical_volume" : true,
    "extend_times" : 0,
    "retention_policy" : "produce_reject",
    "ipv6_enable" : false,
    "ipv6_connect_addresses" : [ ],
    "connector_enable" : false,
    "connector_node_num" : 0,
    "connector_id" : "",
    "rest_enable" : false,
    "rest_connect_address" : "",
    "kafka_public_status" : "closed",
    "public_bandwidth" : 0,
    "message_query_inst_enable" : true,
    "vpc_client_plain" : false,
    "support_features" :
"kafka.new.pod.port,feature.physerver.kafka.topic.modify,feature.physerver.kafka.topic.accesspolicy,message_t
race_enable,features.pod.token.access,feature.physerver.kafka.pulbic.dynamic,roma_app_enable,features.log.c
ollection,auto_topic_switch,feature.physerver.kafka.user.manager",
    "trace_enable" : false,
    "agent_enable" : false,
    "pod_connect_address" : "100.86.75.15:9080,100.86.142.77:9080,100.86.250.167:9080",
    "disk_encrypted" : false,
    "kafka_private_connect_address" : "192.168.0.61:9092,192.168.0.100:9092,192.168.0.72:9092",
```

```
    "enable_auto_topic" : false,
    "new_spec_billing_enable" : false,
    "ces_version" : "linux"
  } ],
  "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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.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
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            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**。

5.1.3 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 5-13 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

状态码：200

表 5-14 响应 Body 参数

参数	参数类型	描述
name	String	实例名称。
engine	String	引擎。
engine_version	String	版本。
description	String	实例描述。
specification	String	实例规格。
storage_space	Integer	消息存储空间，单位：GB。
partition_num	String	Kafka实例的分区数量。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
dns_enable	Boolean	实例是否开启域名访问功能。 ● true：开启 ● false：未开启
connect_address	String	实例连接IP地址。
port	Integer	实例连接端口。
status	String	实例的状态。详细状态说明请参考**实例状态说明**。
instance_id	String	实例ID。
resource_spec_code	String	资源规格标识。 ● dms.instance.kafka.cluster.c3.mini：Kafka实例的基准带宽为100MByte/秒。 ● dms.instance.kafka.cluster.c3.small.2：Kafka实例的基准带宽为300MByte/秒。 ● dms.instance.kafka.cluster.c3.middle.2：Kafka实例的基准带宽为600MByte/秒。 ● dms.instance.kafka.cluster.c3.high.2：Kafka实例的基准带宽为1200MByte/秒。
charging_mode	Integer	付费模式，1表示按需计费，0表示包年/包月计费。

参数	参数类型	描述
vpc_id	String	VPC ID。
vpc_name	String	VPC的名称。
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。
subnet_name	String	子网名称。
subnet_cidr	String	子网网段。
user_id	String	用户ID。
user_name	String	用户名。
access_user	String	实例访问用户名。
order_id	String	订单ID，只有在包周期计费时才会有order_id值，其他计费方式order_id值为空。
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_publicip	Boolean	实例是否开启公网访问功能。 ● true：开启 ● false：未开启
management_connect_address	String	Kafka实例的Kafka Manager连接地址。
ssl_enable	Boolean	是否开启安全认证。 ● true：开启 ● false：未开启
broker_ssl_enable	Boolean	是否开启broker间副本加密传输。 ● true：开启 ● false：未开启

参数	参数类型	描述
kafka_security_protocol	String	Kafka使用的安全协议。 若实例详情中不存在port_protocols返回参数，则kafka_security_protocol同时代表内网访问、公网访问以及跨VPC访问的安全协议。 若实例详情中存在port_protocols返回参数，则kafka_security_protocol仅代表跨VPC访问的安全协议。内网访问公网访问的安全协议请参考port_protocols参数。 ● PLAINTEXT: 既未采用SSL证书进行加密传输，也不支持账号密码认证。性能更好，安全性较低，建议在生产环境下公网访问不使用此方式。 ● SASL_SSL: 采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT: 明文传输，支持账号密码认证，性能更好，建议使用SCRAM-SHA-512机制。
sasl_enabled_mechanisms	Array of strings	开启SASL后使用的认证机制。 ● PLAIN: 简单的用户名密码校验。 ● SCRAM-SHA-512: 用户凭证校验，安全性比PLAIN机制更高。
ssl_two_way_enable	Boolean	是否开启双向认证。
cert_replaced	Boolean	是否能够证书替换。
public_management_connect_address	String	公网访问Kafka Manager连接地址。
enterprise_project_id	String	企业项目ID。
is_logical_volume	Boolean	实例扩容时用于区分老实例与新实例。 ● true：新创建的实例，允许磁盘动态扩容不需要重启。 ● false：老实例
extend_times	Integer	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。
enable_auto_topic	Boolean	是否打开kafka自动创建topic功能。 ● true：开启 ● false：关闭

参数	参数类型	描述
type	String	实例类型：集群，cluster。
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。
available_zones	Array of strings	实例节点所在的可用区，返回“可用区ID”。
available_zone_names	Array of strings	实例节点所在的可用区名称，返回“可用区名称”。
total_storage_space	Integer	总共消息存储空间，单位：GB。
public_connect_address	String	实例公网连接IP地址。当实例开启了公网访问，实例才包含该参数。
public_connect_domain_name	String	实例公网连接域名。当实例开启了公网访问，实例才包含该参数。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	IO规格。
service_type	String	服务类型。
storage_type	String	存储类型。
retention_policy	String	消息老化策略。
kafka_public_status	String	Kafka公网开启状态。
public_bandwidth	Integer	kafka公网访问带宽。
enable_log_collection	Boolean	是否开启消息收集功能。
new_auth_cert	Boolean	是否开启新证书。
cross_vpc_info	String	跨VPC访问信息。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
connector_enable	Boolean	是否开启转储。新规格产品暂不支持开启转储。

参数	参数类型	描述
connector_node_num	Integer	connector节点数量。
connector_id	String	转储任务ID。
rest_enable	Boolean	是否开启Kafka rest功能。
rest_connect_address	String	Kafka rest连接地址。
public_boundwidth	Integer	kafka公网访问带宽。待删除版本。
message_query_inst_enable	Boolean	是否开启消息查询功能。
vpc_client_plain	Boolean	是否开启VPC明文访问。
support_features	String	Kafka实例支持的特性功能。
trace_enable	Boolean	是否开启消息轨迹功能。
agent_enable	Boolean	是否开启代理。
pod_connect_address	String	租户侧连接地址。
disk_encrypted	Boolean	是否开启磁盘加密。
disk_encrypted_key	String	磁盘加密key，未开启磁盘加密时为空。
kafka_private_connect_address	String	Kafka实例内网连接地址。
kafka_private_connect_domain_name	String	Kafka实例内网连接域名。
ces_version	String	云监控版本。
public_access_enabled	String	区分实例什么时候开启的公网访问 取值范围： • true：已开启公网访问 • actived：已开启公网访问 • closed：已关闭公网访问 • false：已关闭公网访问
node_num	Integer	节点数。
port_protocols	**PortProtocolsEntity** object	实例支持的连接方式及其连接地址。
enable_acl	Boolean	是否开启访问控制。

参数	参数类型	描述
new_spec_billing_enable	Boolean	是否启用新规格计费。
broker_num	Integer	节点数量。
tags	Array of **TagEntity** objects	标签列表。
dr_enable	Boolean	是否为容灾实例。

表 5-15 PortProtocolsEntity

参数	参数类型	描述
private_plain_enable	Boolean	实例是否支持内网PLAINTEXT访问接入方式。 ● true：实例支持内网PLAINTEXT访问方式接入方式。 ● false：实例不支持内网PLAINTEXT访问接入方式。
private_plain_address	String	kafka内网PLAINTEXT接入方式连接地址。
private_plain_domain_name	String	内网明文连接域名
private_sasl_ssl_enable	Boolean	实例是否支持内网SASL_SSL访问接入方式。 ● true：实例支持内网SASL_SSL访问方式接入方式。 ● false：实例不支持内网SASL_SSL访问接入方式。
private_sasl_ssl_address	String	kafka内网SASL_SSL接入方式连接地址。
private_sasl_ssl_domain_name	String	内网SASL_SSL连接域名
private_sasl_plaintext_enable	Boolean	实例是否支持内网SASL_PLAINTEXT访问接入方式。 ● true，实例支持内网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持内网SASL_PLAINTEXT访问接入方式。
private_sasl_plaintext_address	String	kafka内网SASL_PLAINTEXT接入方式连接地址。

参数	参数类型	描述
private_sasl_plaintext_domain_name	String	内网SASL_PLAINTEXT连接域名
public_plain_enable	Boolean	实例是否支持公网PLAINTEXT访问接入方式。 ● true，实例支持公网PLAINTEXT访问方式接入方式。 ● false，实例不支持公网PLAINTEXT访问接入方式。
public_plain_address	String	kafka公网PLAINTEXT接入方式连接地址。
public_plain_domain_name	String	公网明文连接域名
public_sasl_ssl_enable	Boolean	实例是否支持公网SASL_SSL访问接入方式。 ● true，实例支持内网SASL_SSL访问方式接入方式。 ● false，实例不支持公网SASL_SSL访问接入方式。
public_sasl_ssl_address	String	kafka公网SASL_SSL接入方式连接地址。
public_sasl_ssl_domain_name	String	公网SASL_SSL连接域名
public_sasl_plaintext_enable	Boolean	实例是否支持公网SASL_PLAINTEXT访问接入方式。 ● true，实例支持公网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持公网SASL_PLAINTEXT访问接入方式。
public_sasl_plaintext_address	String	kafka公网SASL_PLAINTEXT接入方式连接地址。
public_sasl_plaintext_domain_name	String	公网SASL_PLAINTEXT连接域名

表 5-16 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_ . : = + - @ ● 不能以_sys_开头。 ● 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_ . : = + - @

请求示例

查询指定实例

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

响应示例

状态码：200

查询指定实例成功。

```
{
  "name" : "kafka-2085975099",
  "engine" : "kafka",
  "port" : 9092,
  "status" : "RUNNING",
  "type" : "cluster",
  "specification" : "100MB",
  "engine_version" : "1.1.0",
  "connect_address" : "192.168.0.100,192.168.0.61,192.168.0.72",
  "instance_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "resource_spec_code" : "dms.instance.kafka.cluster.c3.mini",
  "charging_mode" : 1,
  "vpc_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "vpc_name" : "dms-test",
  "created_at" : "1585618587087",
  "product_id" : "00300-30308-0--0",
  "security_group_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "security_group_name" : "Sys-default",
  "subnet_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "available_zones" : [ "38b0f7a602344246bcb0da47b5d548e7" ],
  "available_zone_names" : [ "AZ1" ],
  "user_id" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "user_name" : "paas_dms",
  "maintain_begin" : "02:00:00",
```

```
  "maintain_end" : "06:00:00",
  "enable_log_collection" : false,
  "new_auth_cert" : false,
  "storage_space" : 492,
  "total_storage_space" : 600,
  "used_storage_space" : 25,
  "partition_num" : "300",
  "enable_publicip" : false,
  "ssl_enable" : false,
  "broker_ssl_enable" : false,
  "cert_replaced" : false,
  "management_connect_address" : "https://192.168.0.100:9999",
  "cross_vpc_info" : "{\"192.168.0.61\":{\"advertised_ip\":\"192.168.0.61\",\"port\":9011,\"port_id\":\"xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.72\":{\"advertised_ip\":\"192.168.0.72\",\"port\":9011,\"port_id
\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.100\":{\"advertised_ip\":\"192.168.0.100\",\"port
\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"}}",
  "storage_resource_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "storage_spec_code" : "dms.physical.storage.ultra",
  "service_type" : "advanced",
  "storage_type" : "hec",
  "enterprise_project_id" : "0",
  "is_logical_volume" : true,
  "extend_times" : 0,
  "retention_policy" : "produce_reject",
  "ipv6_enable" : false,
  "ipv6_connect_addresses" : [ ],
  "connector_enable" : false,
  "connector_node_num" : 0,
  "connector_id" : "",
  "rest_enable" : false,
  "rest_connect_address" : "",
  "kafka_public_status" : "closed",
  "public_bandwidth" : 0,
  "message_query_inst_enable" : true,
  "vpc_client_plain" : false,
  "support_features" :
"kafka.new.pod.port,feature.physerver.kafka.topic.modify,feature.physerver.kafka.topic.accesspolicy,message_t
race_enable,features.pod.token.access,feature.physerver.kafka.pulbic.dynamic,roma_app_enable,features.log.c
ollection,auto_topic_switch,feature.physerver.kafka.user.manager",
  "trace_enable" : false,
  "agent_enable" : false,
  "pod_connect_address" : "100.86.75.15:9080,100.86.142.77:9080,100.86.250.167:9080",
  "disk_encrypted" : false,
  "kafka_private_connect_address" : "192.168.0.61:9092,192.168.0.100:9092,192.168.0.72:9092",
  "enable_auto_topic" : false,
  "new_spec_billing_enable" : false,
  "ces_version" : "linux",
  "port_protocols" : "{\"private_plain_enable\": true,\"private_plain_address\":
\"192.xxx.xxx.xxx:9092,192.xxx.xxx.xxx:9092,192.xxx.xxx.xxx:9092\",\"private_sasl_ssl_enable\":
true,\"private_sasl_ssl_address\":
\"192.xxx.xxx.xxx:9093,192.xxx.xxx.xxx:9093,192.xxx.xxx.xxx:9093\",\"private_sasl_plaintext_enable\":
false,\"private_sasl_plaintext_address\": \"\",\"public_plain_enable\": true,\"public_plain_address\":
\"100.xxx.xxx.xxx:9094,100.xxx.xxx.xxx:9094,100.xxx.xxx.xxx:9094\",\"public_sasl_ssl_enable\":
true,\"public_sasl_ssl_address\":
\"100.xxx.xxx.xxx:9095,100.xxx.xxx.xxx:9095,100.xxx.xxx.xxx:9095\",\"public_sasl_plaintext_enable\":
false,\"public_sasl_plaintext_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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

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

```
    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            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	查询指定实例成功。

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 5-17 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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            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

修改实例信息。

调用方法

请参见**如何调用API**。

URI

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

表 5-18 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 5-19 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、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"。
maintain_begin	否	String	维护时间窗开始时间，格式为HH:mm:ss。
maintain_end	否	String	维护时间窗结束时间，格式为HH:mm:ss。
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。
retention_policy	否	String	容量阈值策略。 支持两种策略模式： ● produce_reject: 生产受限 ● time_base: 自动删除
enterprise_project_id	否	String	企业项目。

响应参数

无

请求示例

- 修改实例的名称和描述。

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

{
  "name" : "kafka001",
  "description" : "kafka description"
}
```

- 修改实例的名称、描述和维护时间窗。

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

```
{
  "name" : "dms002",
  "description" : "instance description",
  "maintain_begin" : "02:00:00",
  "maintain_end" : "06:00:00"
}
```

- 修改容量阈值策略。

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

{
  "retention_policy" : "time_base"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

- 修改实例的名称和描述。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withDescription("kafka description");
        body.withName("kafka001");
        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());
        }
    }
}
```

- 修改实例的名称、描述和维护时间窗。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withMaintainEnd("06:00:00");
        body.withMaintainBegin("02:00:00");
        body.withDescription("instance description");
        body.withName("dms002");
        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());
        }
    }
}
```

- 修改容量阈值策略。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withRetentionPolicy(UpdateInstanceReq.RetentionPolicyEnum.fromValue("time_base"));
        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

- 修改实例的名称和描述。

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

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
            description="kafka description",
            name="kafka001"
        )
        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)
```

- 修改实例的名称、描述和维护时间窗。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
```



```
            maintain_end="06:00:00",
            maintain_begin="02:00:00",
            description="instance description",
            name="dms002"
        )
        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)
```

- 修改容量阈值策略。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
            retention_policy="time_base"
        )
        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

- 修改实例的名称和描述。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceRequest{}
    request.InstanceId = "{instance_id}"
    descriptionUpdateInstanceReq:= "kafka description"
    nameUpdateInstanceReq:= "kafka001"
    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)
    }
}
```

- 修改实例的名称、描述和维护时间窗。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
```

```
        Build())

    request := &model.UpdateInstanceRequest{}
    request.InstanceId = "{instance_id}"
    maintainEndUpdateInstanceReq:= "06:00:00"
    maintainBeginUpdateInstanceReq:= "02:00:00"
    descriptionUpdateInstanceReq:= "instance description"
    nameUpdateInstanceReq:= "dms002"
    request.Body = &model.UpdateInstanceReq{
        MaintainEnd: &maintainEndUpdateInstanceReq,
        MaintainBegin: &maintainBeginUpdateInstanceReq,
        Description: &descriptionUpdateInstanceReq,
        Name: &nameUpdateInstanceReq,
    }
    response, err := client.UpdateInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 修改容量阈值策略。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceRequest{}
    request.InstanceId = "{instance_id}"
    retentionPolicyUpdateInstanceReq:= model.GetUpdateInstanceReqRetentionPolicyEnum().TIME_BASE
    request.Body = &model.UpdateInstanceReq{
        RetentionPolicy: &retentionPolicyUpdateInstanceReq,
    }
    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 批量重启或删除实例

功能介绍

批量重启或删除实例。

在实例重启过程中，客户端的生产与消费消息等请求会被拒绝。

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 5-20 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**。

请求参数

表 5-21 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：restart、delete
all_failure	否	String	参数值为kafka，表示删除租户所有创建失败的Kafka实例。

响应参数

状态码：200

表 5-22 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 5-23 results

参数	参数类型	描述
result	String	操作结果。 • success: 操作成功 • failed: 操作失败
instance	String	实例ID。

请求示例

- 批量重启实例。

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

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

- 批量删除实例。

```
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实例。

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

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

响应示例

状态码：200

批量重启或删除实例成功。

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

SDK 代码示例

SDK代码示例如下。

Java

- 批量重启实例。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class BatchRestartOrDeleteInstancesSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
```

```
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchRestartOrDeleteInstancesRequest request = new BatchRestartOrDeleteInstancesRequest();
        BatchRestartOrDeleteInstanceReq body = new BatchRestartOrDeleteInstanceReq();
        List<String> listbodyInstances = new ArrayList<>();
        listbodyInstances.add("54602a9d-5e22-4239-9123-77e350df4a34");
        listbodyInstances.add("7166cdea-dbad-4d79-9610-7163e6f8b640");
        body.withAction(BatchRestartOrDeleteInstanceReq.ActionEnum.fromValue("restart"));
        body.withInstances(listbodyInstances);
        request.withBody(body);
        try {
            BatchRestartOrDeleteInstancesResponse response =
client.batchRestartOrDeleteInstances(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());
        }
    }
}
```

- 批量删除实例。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class BatchRestartOrDeleteInstancesSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchRestartOrDeleteInstancesRequest request = new BatchRestartOrDeleteInstancesRequest();
        BatchRestartOrDeleteInstanceReq body = new BatchRestartOrDeleteInstanceReq();
        []string listbodyInstances = new ArrayList<>();
        listbodyInstances.add("54602a9d-5e22-4239-9123-77e350df4a34");
        listbodyInstances.add("7166cdea-dbad-4d79-9610-7163e6f8b640");
```

```
        body.withAction(BatchRestartOrDeleteInstanceReq.ActionEnum.fromValue("delete"));
        body.withInstances(listbodyInstances);
        request.withBody(body);
        try {
            BatchRestartOrDeleteInstancesResponse response =
client.batchRestartOrDeleteInstances(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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class BatchRestartOrDeleteInstancesSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchRestartOrDeleteInstancesRequest request = new BatchRestartOrDeleteInstancesRequest();
        BatchRestartOrDeleteInstanceReq body = new BatchRestartOrDeleteInstanceReq();
        body.withAllFailure(BatchRestartOrDeleteInstanceReq.AllFailureEnum.fromValue("kafka"));
        body.withAction(BatchRestartOrDeleteInstanceReq.ActionEnum.fromValue("delete"));
        request.withBody(body);
        try {
            BatchRestartOrDeleteInstancesResponse response =
client.batchRestartOrDeleteInstances(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

- 批量重启实例。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchRestartOrDeleteInstancesRequest()
        listInstancesbody = [
            "54602a9d-5e22-4239-9123-77e350df4a34",
            "7166cdea-dbad-4d79-9610-7163e6f8b640"
        ]
        request.body = BatchRestartOrDeleteInstanceReq(
            action="restart",
            instances=listInstancesbody
        )
        response = client.batch_restart_or_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)
```

- 批量删除实例。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchRestartOrDeleteInstancesRequest()
        listInstancesbody = [
            "54602a9d-5e22-4239-9123-77e350df4a34",
            "7166cdea-dbad-4d79-9610-7163e6f8b640"
        ]
        request.body = BatchRestartOrDeleteInstanceReq(
            action="delete",
            instances=listInstancesbody
        )
        response = client.batch_restart_or_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实例。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchRestartOrDeleteInstancesRequest()
        request.body = BatchRestartOrDeleteInstanceReq(
            all_failure="kafka",
            action="delete"
        )
        response = client.batch_restart_or_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

- 批量重启实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchRestartOrDeleteInstancesRequest{}
    var listInstancesbody = List{
        "54602a9d-5e22-4239-9123-77e350df4a34",
        "7166cdea-dbad-4d79-9610-7163e6f8b640",
    }
    request.Body = &model.BatchRestartOrDeleteInstanceReq{
        Action: model.GetBatchRestartOrDeleteInstanceReqActionEnum().RESTART,
        Instances: &listInstancesbody,
    }
    response, err := client.BatchRestartOrDeleteInstances(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 批量删除实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

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

- 删除所有创建失败的实例。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
```

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

    request := &model.BatchRestartOrDeleteInstancesRequest{}
    allFailureBatchRestartOrDeleteInstanceReq:=
model.GetBatchRestartOrDeleteInstanceReqAllFailureEnum().KAFKA
    request.Body = &model.BatchRestartOrDeleteInstanceReq{
        AllFailure: &allFailureBatchRestartOrDeleteInstanceReq,
        Action: model.GetBatchRestartOrDeleteInstanceReqActionEnum().DELETE,
    }
    response, err := client.BatchRestartOrDeleteInstances(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	批量重启或删除实例成功。
204	删除所有创建失败的Kafka实例成功。

错误码

请参见**错误码**。

5.1.7 获取实例配置

功能介绍

获取实例配置。

调用方法

请参见**如何调用API**。

URI

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

表 5-24 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 5-25 响应 Body 参数

参数	参数类型	描述
kafka_configs	Array of **InstanceConfig** objects	kafka配置列表。

表 5-26 InstanceConfig

参数	参数类型	描述
name	String	配置名称。
valid_values	String	有效值。
default_value	String	默认值。
config_type	String	配置类型：static/dynamic。
value	String	配置当前值。
value_type	String	值类型。

请求示例

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

响应示例

状态码：200

获取实例配置成功。

```
{
  "kafka_configs" : [ {
```

```
  "name" : "min.insync.replicas",
  "valid_values" : "1~3",
  "default_value" : "1",
  "config_type" : "dynamic",
  "value" : "1",
  "value_type" : "integer"
}, {
  "name" : "message.max.bytes",
  "valid_values" : "0~10485760",
  "default_value" : "10485760",
  "config_type" : "dynamic",
  "value" : "10485760",
  "value_type" : "integer"
}, {
  "name" : "auto.create.groups.enable",
  "valid_values" : "true,false",
  "default_value" : "true",
  "config_type" : "dynamic",
  "value" : "true",
  "value_type" : "enum"
}, {
  "name" : "connections.max.idle.ms",
  "valid_values" : "5000~600000",
  "default_value" : "600000",
  "config_type" : "static",
  "value" : "600000",
  "value_type" : "integer"
}, {
  "name" : "log.retention.hours",
  "valid_values" : "1~168",
  "default_value" : "72",
  "config_type" : "static",
  "value" : "72",
  "value_type" : "integer"
}, {
  "name" : "max.connections.per.ip",
  "valid_values" : "100~20000",
  "default_value" : "1000",
  "config_type" : "dynamic",
  "value" : "1000",
  "value_type" : "integer"
}, {
  "name" : "group.max.session.timeout.ms",
  "valid_values" : "6000~1800000",
  "default_value" : "1800000",
  "config_type" : "static",
  "value" : "1800000",
  "value_type" : "integer"
}, {
  "name" : "unclean.leader.election.enable",
  "valid_values" : "true,false",
  "default_value" : "false",
  "config_type" : "dynamic",
  "value" : "false",
  "value_type" : "enum"
}, {
  "name" : "default.replication.factor",
  "valid_values" : "1~3",
  "default_value" : "3",
  "config_type" : "static",
  "value" : "3",
  "value_type" : "integer"
}, {
  "name" : "offsets.retention.minutes",
  "valid_values" : "1440~30240",
  "default_value" : "20160",
  "config_type" : "dynamic",
  "value" : "20160",
  "value_type" : "integer"
```

```
 }, {
   "name" : "num.partitions",
   "valid_values" : "1~200",
   "default_value" : "3",
   "config_type" : "static",
   "value" : "3",
   "value_type" : "integer"
 }, {
   "name" : "group.min.session.timeout.ms",
   "valid_values" : "6000~300000",
   "default_value" : "6000",
   "config_type" : "static",
   "value" : "6000",
   "value_type" : "integer"
 }, {
   "name" : "allow.everyone.if.no.acl.found",
   "valid_values" : "true,false",
   "default_value" : "true",
   "config_type" : "static",
   "value" : "true",
   "value_type" : "enum"
 } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowInstanceConfigsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowInstanceConfigsRequest request = new ShowInstanceConfigsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowInstanceConfigsResponse response = client.showInstanceConfigs(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowInstanceConfigsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_instance_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowInstanceConfigsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowInstanceConfigs(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	获取实例配置成功。

错误码

请参见**错误码**。

5.1.8 修改实例配置

功能介绍

修改实例配置。

调用方法

请参见**如何调用API**。

URI

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

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。

请求参数

表 5-28 请求 Body 参数

参数	是否必选	参数类型	描述
kafka_configs	否	Array of **ModifyInstanceConfig** objects	kafka待修改配置列表。

表 5-29 ModifyInstanceConfig

参数	是否必选	参数类型	描述
name	否	String	修改的配置名称。
value	否	String	配置的修改值。

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	配置修改任务ID。
dynamic_config	Integer	本次修改动态配置参数个数。
static_config	Integer	本次修改静态配置参数个数。

请求示例

修改实例的连接空闲超时和日志删除的时间阈值。

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

{
```

```
 "kafka_configs" : [ {
   "name" : "connections.max.idle.ms",
   "value" : "500000"
 }, {
   "name" : "log.retention.hours",
   "value" : "66"
 } ]
}
```

响应示例

状态码：200

修改实例配置成功。

```
{
 "job_id" : "8abfa7b38ba79a20018ba9afc550576a",
 "dynamic_config" : 0,
 "static_config" : 2
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改实例的连接空闲超时和日志删除的时间阈值。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class ModifyInstanceConfigsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ModifyInstanceConfigsRequest request = new ModifyInstanceConfigsRequest();
        request.withInstanceId("{instance_id}");
```

```
        ModifyInstanceConfigsReq body = new ModifyInstanceConfigsReq();
        List<ModifyInstanceConfig> listbodyKafkaConfigs = new ArrayList<>();
        listbodyKafkaConfigs.add(
            new ModifyInstanceConfig()
                .withName("connections.max.idle.ms")
                .withValue("500000")
        );
        listbodyKafkaConfigs.add(
            new ModifyInstanceConfig()
                .withName("log.retention.hours")
                .withValue("66")
        );
        body.withKafkaConfigs(listbodyKafkaConfigs);
        request.withBody(body);
        try {
            ModifyInstanceConfigsResponse response = client.modifyInstanceConfigs(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

修改实例的连接空闲超时和日志删除的时间阈值。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ModifyInstanceConfigsRequest()
        request.instance_id = "{instance_id}"
        listKafkaConfigsbody = [
            ModifyInstanceConfig(
                name="connections.max.idle.ms",
                value="500000"
            ),
            ModifyInstanceConfig(
```

```
            name="log.retention.hours",
            value="66"
        )
    ]
    request.body = ModifyInstanceConfigsReq(
        kafka_configs=listKafkaConfigsbody
    )
    response = client.modify_instance_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

修改实例的连接空闲超时和日志删除的时间阈值。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ModifyInstanceConfigsRequest{}
    request.InstanceId = "{instance_id}"
    nameKafkaConfigs:= "connections.max.idle.ms"
    valueKafkaConfigs:= "500000"
    nameKafkaConfigs1:= "log.retention.hours"
    valueKafkaConfigs1:= "66"
    var listKafkaConfigsbody = []model.ModifyInstanceConfig{
        {
            Name: &nameKafkaConfigs,
            Value: &valueKafkaConfigs,
        },
        {
            Name: &nameKafkaConfigs1,
            Value: &valueKafkaConfigs1,
        },
    }
    request.Body = &model.ModifyInstanceConfigsReq{
        KafkaConfigs: &listKafkaConfigsbody,
```

```
    }
    response, err := client.ModifyInstanceConfigs(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	修改实例配置成功。

错误码

请参见**错误码**。

5.1.9 实例升级

功能介绍

实例内核升级

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/kafka/instances/{instance_id}/upgrade

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
is_schedule	否	Boolean	是否作为定时任务执行。若非定时执行，is_schedule和execute_at字段可为空。若为定时执行，is_schedule为true，execute_at字段非空。
execute_at	否	Long	定时时间，格式为Unix时间戳，单位为毫秒

响应参数

状态码：200

表 5-33 响应 Body 参数

参数	参数类型	描述
-	String	提交升级任务id

状态码：400

表 5-34 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

请求示例

升级id为instance_id的实例。

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

{
  "is_schedule" : true,
  "execute_at" : 1695796358970
}
```

响应示例

状态码：200

升级成功。

```
93b94287-728d-4bb1-a158-cb66cb0854e7
```

状态码

状态码	描述
200	升级成功。
400	升级失败。

错误码

请参见**错误码**。

5.2 实例管理

5.2.1 重置密码

功能介绍

重置密码（只针对开通SSL的实例）。

调用方法

请参见**如何调用API**。

URI

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

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。

请求参数

表 5-36 请求 Body 参数

参数	是否必选	参数类型	描述
new_password	是	String	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`~!@#$%^&*()-_=+|[{}];:'",<.>/? 和空格，并且不能以-开头。

响应参数

无

请求示例

重置密码。

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

{
  "new_password" : "********"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

重置密码。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetPasswordSolution {
```

```
    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetPasswordRequest request = new ResetPasswordRequest();
        request.withInstanceId("{instance_id}");
        ResetPasswordReq body = new ResetPasswordReq();
        body.withNewPassword("********");
        request.withBody(body);
        try {
            ResetPasswordResponse response = client.resetPassword(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

重置密码。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
```

```
        .build()

    try:
        request = ResetPasswordRequest()
        request.instance_id = "{instance_id}"
        request.body = ResetPasswordReq(
            new_password="********"
        )
        response = client.reset_password(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

重置密码。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetPasswordRequest{}
    request.InstanceId = "{instance_id}"
    request.Body = &model.ResetPasswordReq{
        NewPassword: "********",
    }
    response, err := client.ResetPassword(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	重置密码成功。

错误码

请参见**错误码**。

5.2.2 重置 Manager 密码

功能介绍

重置Manager密码。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/kafka-manager-password

表 5-37 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 5-38 请求 Body 参数

参数	是否必选	参数类型	描述
new_password	否	String	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`~!@#$%^&*()-_=+|[{}];:'",<.>/? 和空格，并且不能以-开头。

响应参数

无

请求示例

重置Kafka Manager的密码。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/kafka-manager-password

{
  "new_password" : "********"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

重置Kafka Manager的密码。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetManagerPasswordSolution {
```

```
    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetManagerPasswordRequest request = new ResetManagerPasswordRequest();
        request.withInstanceId("{instance_id}");
        ResetManagerPasswordReq body = new ResetManagerPasswordReq();
        body.withNewPassword("********");
        request.withBody(body);
        try {
            ResetManagerPasswordResponse response = client.resetManagerPassword(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Kafka Manager的密码。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
```

```
        .build()

    try:
        request = ResetManagerPasswordRequest()
        request.instance_id = "{instance_id}"
        request.body = ResetManagerPasswordReq(
            new_password="********"
        )
        response = client.reset_manager_password(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Kafka Manager的密码。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetManagerPasswordRequest{}
    request.InstanceId = "{instance_id}"
    newPasswordResetManagerPasswordReq:= "********"
    request.Body = &model.ResetManagerPasswordReq{
        NewPassword: &newPasswordResetManagerPasswordReq,
    }
    response, err := client.ResetManagerPassword(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	重置密码成功。

错误码

请参见**错误码**。

5.2.3 重启 Manager

功能介绍

重启Manager。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/restart-kafka-manager

表 5-39 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 5-40 响应 Body 参数

参数	参数类型	描述
result	String	执行结果。
instance_id	String	实例ID。

请求示例

重启Kafka Manager。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/restart-kafka-manager
```

响应示例

状态码：200

重启Manager成功。

```
{
  "result" : "success",
  "instance_id" : "605cd78c-92dc-4335-8bae-43677f31fd6c"
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class RestartManagerSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
```

```
            .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
            .build();
        RestartManagerRequest request = new RestartManagerRequest();
        request.withInstanceId("{instance_id}");
        try {
            RestartManagerResponse response = client.restartManager(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = RestartManagerRequest()
        request.instance_id = "{instance_id}"
        response = client.restart_manager(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.RestartManagerRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.RestartManager(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	重启Manager成功。

错误码

请参见**错误码**。

5.2.4 开启或关闭实例自动创建 topic 功能

功能介绍

开启或关闭实例自动创建topic功能。

调用方法

请参见**如何调用API**。

URI

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

表 5-41 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 5-42 请求 Body 参数

参数	是否必选	参数类型	描述
enable_auto_topic	是	Boolean	是否开启自动创建topic功能。

响应参数

无

请求示例

开启实例自动创建topic功能。

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

{
  "enable_auto_topic" : true
}
```

响应示例

无

状态码

状态码	描述
200	开启或关闭成功。

错误码

请参见**错误码**。

5.2.5 修改实例跨 VPC 访问的内网 IP

功能介绍

修改实例跨VPC访问的内网IP。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/crossvpc/modify

表 5-43 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 5-44 请求 Body 参数

参数	是否必选	参数类型	描述
advertised_ip_contents	是	Map<String,String>	用户自定义的advertised_ip_contents键值对。 键是listeners IP。 值是advertised.listeners IP，或者域名。 **说明** IP修改未修改项也需填上。

响应参数

状态码：200

表 5-45 响应 Body 参数

参数	参数类型	描述
success	Boolean	修改跨VPC访问结果。

参数	参数类型	描述
results	Array of **results** objects	修改broker跨VPC访问的结果列表。

表 5-46 results

参数	参数类型	描述
advertised_ip	String	advertised.listeners IP/域名。
success	Boolean	修改broker跨VPC访问的状态。
ip	String	listeners IP。

请求示例

修改实例跨VPC访问的内网IP。

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

{
  "advertised_ip_contents" : {
    "192.168.245.246" : "192.168.245.247",
    "192.168.197.36" : "192.168.197.38",
    "192.168.190.11" : "192.168.190.11"
  }
}
```

响应示例

状态码：200

修改实例跨VPC访问的内网IP成功。

```
{
  "success" : true,
  "results" : [ {
    "advertised_ip" : "192.168.197.36",
    "success" : true,
    "ip" : "192.168.197.36"
  }, {
    "advertised_ip" : "192.168.190.11",
    "success" : true,
    "ip" : "192.168.190.11"
  }, {
    "advertised_ip" : "192.168.245.255",
    "success" : true,
    "ip" : "192.168.245.246"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改实例跨VPC访问的内网IP。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

import java.util.Map;
import java.util.HashMap;

public class UpdateInstanceCrossVpcIpSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceCrossVpcIpRequest request = new UpdateInstanceCrossVpcIpRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceCrossVpcIpReq body = new UpdateInstanceCrossVpcIpReq();
        Map<String, String> listbodyAdvertisedIpContents = new HashMap<>();
        listbodyAdvertisedIpContents.put("192.168.245.246", "192.168.245.247");
        listbodyAdvertisedIpContents.put("192.168.197.36", "192.168.197.38");
        listbodyAdvertisedIpContents.put("192.168.190.11", "192.168.190.11");
        body.withAdvertisedIpContents(listbodyAdvertisedIpContents);
        request.withBody(body);
        try {
            UpdateInstanceCrossVpcIpResponse response = client.updateInstanceCrossVpcIp(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

修改实例跨VPC访问的内网IP。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceCrossVpcIpRequest()
        request.instance_id = "{instance_id}"
        listAdvertisedIpContentsbody = {
            "192.168.245.246": "192.168.245.247",
            "192.168.197.36": "192.168.197.38",
            "192.168.190.11": "192.168.190.11"
        }
        request.body = UpdateInstanceCrossVpcIpReq(
            advertised_ip_contents=listAdvertisedIpContentsbody
        )
        response = client.update_instance_cross_vpc_ip(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

修改实例跨VPC访问的内网IP。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceCrossVpcIpRequest{}
    request.InstanceId = "{instance_id}"
    var listAdvertisedIpContentsbody = map[string]string{
        "192.168.245.246": "192.168.245.247",
        "192.168.197.36": "192.168.197.38",
        "192.168.190.11": "192.168.190.11",
    }
    request.Body = &model.UpdateInstanceCrossVpcIpReq{
        AdvertisedIpContents: listAdvertisedIpContentsbody,
    }
    response, err := client.UpdateInstanceCrossVpcIp(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	修改实例跨VPC访问的内网IP成功。

错误码

请参见**错误码**。

5.2.6 查询 Kafka 集群元数据信息

功能介绍

查询Kafka集群元数据信息。

调用方法

请参见**如何调用API**。

URI

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

表 5-47 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 5-48 响应 Body 参数

参数	参数类型	描述
cluster	**cluster** object	集群基本信息。

表 5-49 cluster

参数	参数类型	描述
controller	String	控制器ID。
brokers	Array of **brokers** objects	节点列表。
topics_count	Integer	主题数量。
partitions_count	Integer	分区数量。
online_partitions_count	Integer	在线分区数量。
replicas_count	Integer	副本数量。
isr_replicas_count	Integer	ISR（In-Sync Replicas）副本总数。
consumers_count	Integer	消费组数量。

表 5-50 brokers

参数	参数类型	描述
host	String	节点IP。

参数	参数类型	描述
port	Integer	端口号。
broker_id	String	节点ID。
is_controller	Boolean	是否为contoller节点。
version	String	服务端版本。
register_time	Long	broker注册时间，为unix时间戳格式。
is_health	Boolean	Kafka实例节点的连通性是否正常。

请求示例

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

响应示例

状态码：200

查询Kafka集群元数据信息成功。

```
{
  "cluster" : {
    "controller" : "2",
    "brokers" : [ {
      "host" : "192.168.0.159",
      "port" : 9093,
      "broker_id" : "0",
      "is_controller" : false,
      "version" : "1.1.0",
      "register_time" : 1588754647872,
      "is_health" : true
    }, {
      "host" : "192.168.0.48",
      "port" : 9093,
      "broker_id" : "1",
      "is_controller" : false,
      "version" : "1.1.0",
      "register_time" : 1588754647653,
      "is_health" : true
    }, {
      "host" : "192.168.0.212",
      "port" : 9093,
      "broker_id" : "2",
      "is_controller" : true,
      "version" : "1.1.0",
      "register_time" : 1588754647284,
      "is_health" : true
    } ],
    "topics_count" : 3,
    "partitions_count" : 9,
    "online_partitions_count" : 9,
    "replicas_count" : 27,
    "isr_replicas_count" : 27,
    "consumers_count" : 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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowClusterSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowClusterRequest request = new ShowClusterRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowClusterResponse response = client.showCluster(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowClusterRequest()
        request.instance_id = "{instance_id}"
        response = client.show_cluster(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowClusterRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowCluster(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Kafka集群元数据信息成功。

错误码

请参见**错误码**。

5.2.7 查询 Kafka 实例的协调器信息

功能介绍

查询Kafka实例的协调器信息。

调用方法

请参见**如何调用API**。

URI

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

表 5-51 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 5-52 响应 Body 参数

参数	参数类型	描述
coordinators	Array of **coordinators** objects	所有消费组对应的协调器列表。

表 5-53 coordinators

参数	参数类型	描述
group_id	String	消费组ID。
id	Integer	对应协调器的broker id。
host	String	对应协调器的地址。
port	Integer	端口号。

请求示例

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

响应示例

状态码：200

查询Kafka实例的协调器信息成功。

```
{
  "coordinators" : [ {
    "group_id" : "XXXX",
    "id" : 2,
    "host" : "172.31.1.15",
    "port" : 9091
  }, {
    "group_id" : "XXXX",
    "id" : 2,
    "host" : "172.31.1.15",
    "port" : 9092
  }, {
    "group_id" : "XXXX",
    "id" : 2,
    "host" : "172.31.1.15",
    "port" : 9092
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowCoordinatorsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowCoordinatorsRequest request = new ShowCoordinatorsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowCoordinatorsResponse response = client.showCoordinators(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"
```

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

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

    try:
        request = ShowCoordinatorsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_coordinators(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowCoordinatorsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowCoordinators(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Kafka实例的协调器信息成功。

错误码

请参见**错误码**。

5.2.8 修改 Kafka 实例 Topic 分区的副本

功能介绍

修改Kafka实例Topic分区的副本。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/management/topics/{topic}/replicas-reassignment

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。
topic	是	String	Topic名称。

请求参数

表 5-55 请求 Body 参数

参数	是否必选	参数类型	描述
partitions	否	Array of **partitions** objects	期望调整的分区副本分配情况。

表 5-56 partitions

参数	是否必选	参数类型	描述
partition	否	Integer	分区ID。
replicas	否	Array of integers	副本期望所在的broker ID。其中Array首位为leader副本，所有分区需要有同样数量的副本，副本数不能大于总broker的数量。

响应参数

无

请求示例

修改Topic分区的副本分布位置，分区1的副本分布在broker 1和broker 2，Leader副本在broker 1。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/management/topics/{topic}/replicas-reassignment

{
  "partitions" : [ {
    "partition" : 1,
    "replicas" : [ 1, 2 ]
  }, {
    "partition" : 0,
    "replicas" : [ 0, 1 ]
  } ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改Topic分区的副本分布位置，分区1的副本分布在broker 1和broker 2，Leader副本在broker 1。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;
```

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

public class UpdateTopicReplicaSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateTopicReplicaRequest request = new UpdateTopicReplicaRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        ResetReplicaReq body = new ResetReplicaReq();
        List<Integer> listPartitionsReplicas = new ArrayList<>();
        listPartitionsReplicas.add(0);
        listPartitionsReplicas.add(1);
        List<Integer> listPartitionsReplicas1 = new ArrayList<>();
        listPartitionsReplicas1.add(1);
        listPartitionsReplicas1.add(2);
        List<ResetReplicaReqPartitions> listbodyPartitions = new ArrayList<>();
        listbodyPartitions.add(
            new ResetReplicaReqPartitions()
                .withPartition(1)
                .withReplicas(listPartitionsReplicas1)
        );
        listbodyPartitions.add(
            new ResetReplicaReqPartitions()
                .withPartition(0)
                .withReplicas(listPartitionsReplicas)
        );
        body.withPartitions(listbodyPartitions);
        request.withBody(body);
        try {
            UpdateTopicReplicaResponse response = client.updateTopicReplica(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Topic分区的副本分布位置，分区1的副本分布在broker 1和broker 2，Leader副本在broker 1。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateTopicReplicaRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        listReplicasPartitions = [
            0,
            1
        ]
        listReplicasPartitions1 = [
            1,
            2
        ]
        listPartitionsbody = [
            ResetReplicaReqPartitions(
                partition=1,
                replicas=listReplicasPartitions1
            ),
            ResetReplicaReqPartitions(
                partition=0,
                replicas=listReplicasPartitions
            )
        ]
        request.body = ResetReplicaReq(
            partitions=listPartitionsbody
        )
        response = client.update_topic_replica(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Topic分区的副本分布位置，分区1的副本分布在broker 1和broker 2，Leader副本在broker 1。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateTopicReplicaRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    var listReplicasPartitions = List<Integer>{
        int32(0),
        int32(1),
    }
    var listReplicasPartitions1 = List<Integer>{
        int32(1),
        int32(2),
    }
    partitionPartitions:= int32(1)
    partitionPartitions1:= int32(0)
    var listPartitionsbody = []model.ResetReplicaReqPartitions{
        {
            Partition: &partitionPartitions,
            Replicas: &listReplicasPartitions1,
        },
        {
            Partition: &partitionPartitions1,
            Replicas: &listReplicasPartitions,
        },
    }
    request.Body = &model.ResetReplicaReq{
        Partitions: &listPartitionsbody,
    }
    response, err := client.UpdateTopicReplica(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	调整副本操作成功。

错误码

请参见**错误码**。

5.2.9 修改 Kafka 的接入方式

功能介绍

修改Kafka的内网或者公网接入方式。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/{engine}/instances/{instance_id}/plain-ssl-switch

表 5-57 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**。
engine	是	String	消息引擎。
instance_id	是	String	实例ID。

请求参数

表 5-58 请求 Body 参数

参数	是否必选	参数类型	描述
protocol	否	String	需要开启或者关闭的接入方式。
enable	否	Boolean	• true：开启指定的接入方式。 • false：关闭指定的接入方式。

参数	是否必选	参数类型	描述
user_name	否	String	首次开启SASL时，需要输入用户名。实例创建后，关闭SASL并不会删除已经创建的用户，再次开启SASL时无需传入用户名，传入的用户名将无效。
pass_word	否	String	首次开启SASL时，需要输入用户名的密码。
sasl_enabled_mechanisms	否	Array of strings	开启SASL后使用的认证机制。仅在第一次开启SASL时传入生效。生效后再次传入无效。 ● PLAIN：简单的用户名密码校验。 ● SCRAM-SHA-512：用户凭证校验，安全性比PLAIN机制更高。

响应参数

状态码：200

表 5-59 响应 Body 参数

参数	参数类型	描述
job_id	String	后台任务id。
protocol	String	开启或者关闭的Kafka接入方式。
enable	Boolean	开启动作或者关闭动作。

请求示例

实例第一次开启内网SASL_SSL，需要传入用户名及密码。

```
POST https://{endpoint}/v2/{project_id}/{engine}/instances/{instance_id}/plain-ssl-switch

{
  "protocol" : "private_sasl_ssl_enable",
  "enable" : true,
  "user_name" : "root",
  "pass_word" : "password",
  "sasl_enabled_mechanisms" : [ "SCRAM-SHA-512", "PLAIN" ]
}
```

响应示例

无

状态码

状态码	描述
200	提交修改Kafka接入方式成功。

错误码

请参见**错误码**。

5.2.10 查询 topic 的磁盘存储情况

功能介绍

查询topic在Broker上磁盘占用情况。

调用方法

请参见**如何调用API**。

URI

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

表 5-60 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 5-61 Query 参数

参数	是否必选	参数类型	描述
minSize	否	String	占用磁盘大小，默认值1G (1K，1M，1G)。
top	否	String	占用磁盘大小，查询top N。
percentage	否	String	占用磁盘大小，查询大于占比的分区。

请求参数

无

响应参数

状态码：200

表 5-62 响应 Body 参数

参数	参数类型	描述
broker_list	Array of **DiskusageEntity** objects	Broker列表。

表 5-63 DiskusageEntity

参数	参数类型	描述
broker_name	String	Broker名称。
data_disk_size	String	磁盘容量。
data_disk_use	String	已使用的磁盘容量。
data_disk_free	String	剩余可用的磁盘容量。
data_disk_use_percentage	String	消息标签。
status	String	消息标签。
topic_list	Array of **DiskusageTopicEntity** objects	topic磁盘容量使用列表。

表 5-64 DiskusageTopicEntity

参数	参数类型	描述
size	String	磁盘使用量。
topic_name	String	topic名称。
topic_partition	String	分区。
percentage	Double	磁盘使用量的占比。

请求示例

查询Topic的磁盘存储情况

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

响应示例

状态码：200

查询成功。


```
{
  "broker_list" : [ {
    "broker_name" : "broker-0",
    "data_disk_size" : "66G",
    "data_disk_use" : "53M",
    "data_disk_free" : "63G",
    "data_disk_use_percentage" : "1",
    "status" : "Success get info",
    "topic_list" : [ {
      "size" : "12K",
      "topic_name" : "topic-test",
      "topic_partition" : "2",
      "percentage" : 1.7339533025568183E-5
    }, {
      "size" : "12K",
      "topic_name" : "__consumer_offsets",
      "topic_partition" : "4",
      "percentage" : 1.7339533025568183E-5
    }, {
      "size" : "12K",
      "topic_name" : "__consumer_offsets",
      "topic_partition" : "3",
      "percentage" : 1.7339533025568183E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "__trace",
      "topic_partition" : "6",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "__trace",
      "topic_partition" : "4",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "__trace",
      "topic_partition" : "2",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "__trace",
      "topic_partition" : "0",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "topic-test",
      "topic_partition" : "0",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "topic-1568537362",
      "topic_partition" : "2",
      "percentage" : 1.1559688683712121E-5
    }, {
      "size" : "8.0K",
      "topic_name" : "__consumer_offsets",
      "topic_partition" : "7",
      "percentage" : 1.1559688683712121E-5
    } ]
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowKafkaTopicPartitionDiskusageSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowKafkaTopicPartitionDiskusageRequest request = new ShowKafkaTopicPartitionDiskusageRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowKafkaTopicPartitionDiskusageResponse response = client.showKafkaTopicPartitionDiskusage(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowKafkaTopicPartitionDiskusageRequest()
        request.instance_id = "{instance_id}"
        response = client.show_kafka_topic_partition_diskusage(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowKafkaTopicPartitionDiskusageRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowKafkaTopicPartitionDiskusage(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**。

5.2.11 Kafka 实例开始分区平衡任务

功能介绍

该接口用于向Kafka实例提交分区平衡任务或计算分区平衡预估时间。

调用方法

请参见**如何调用API**。

URI

POST /v2/kafka/{project_id}/instances/{instance_id}/reassign

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。

请求参数

表 5-66 请求 Body 参数

参数	是否必选	参数类型	描述
reassignments	是	Array of **PartitionReassignEntity** objects	分区平衡分配方案。
throttle	否	Integer	分区平衡门限值。
is_schedule	否	Boolean	是否作为定时任务执行。若非定时执行，is_schedule和execute_at字段可为空。若为定时执行，is_schedule为true，execute_at字段非空。
execute_at	否	Long	定时时间，格式为Unix时间戳，单位为毫秒
time_estimate	否	Boolean	设为true表示执行时间预估任务，false为执行分区平衡任务。

表 5-67 PartitionReassignEntity

参数	是否必选	参数类型	描述
topic	是	String	topic名称
brokers	否	Array of integers	分区平衡到的broker列表，自动生成分配方案时需指定该参数。
replication_factor	否	Integer	副本因子，自动生成分配方案时可指定。
assignment	否	Array of **TopicAssignment** objects	手动指定的分配方案。brokers参数与该参数不能同时为空。

表 5-68 TopicAssignment

参数	是否必选	参数类型	描述
partition	否	Integer	手动指定分配方案时的分区号。
partition_brokers	否	Array of integers	手动指定某个分区将要分配的broker列表

响应参数

状态码：200

表 5-69 响应 Body 参数

参数	参数类型	描述
job_id	String	任务ID（当执行分区平衡任务时仅返回job_id）。
reassignment_time	Integer	预估时间，单位为秒（当执行预估时间任务时仅返回reassignment_time）。

请求示例

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

{
  "reassignments" : [ {
    "topic" : "topic-1513476102",
    "brokers" : [ 0, 1, 2 ],
    "replication_factor" : 3,
    "assignment" : [ {
      "partition" : 0,
      "partition_brokers" : [ 0, 1, 2 ]
    }, {
      "partition" : 1,
      "partition_brokers" : [ 1, 2, 0 ]
    }, {
      "partition" : 2,
      "partition_brokers" : [ 2, 0, 1 ]
    } ]
  }, {
    "topic" : "topic-1513558717",
    "brokers" : [ 0, 1, 4 ],
    "replication_factor" : 3,
    "assignment" : [ {
      "partition" : 0,
      "partition_brokers" : [ 0, 1, 2 ]
    }, {
      "partition" : 1,
      "partition_brokers" : [ 1, 2, 0 ]
    }, {
      "partition" : 2,
      "partition_brokers" : [ 2, 0, 1 ]
    } ]
  } ],
  "throttle" : 10000000,
  "time_estimate" : false
}
```

响应示例

状态码：200

提交分区平衡任务成功（若为预估时间任务则返回预估时间）。

```
{
  "job_id" : "8a2c259182ab0e9d0182ab1882560009",
  "reassignment_time" : 10
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class CreateReassignmentTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateReassignmentTaskRequest request = new CreateReassignmentTaskRequest();
        request.withInstanceId("{instance_id}");
        PartitionReassignRequest body = new PartitionReassignRequest();
        List<Integer> listAssignmentPartitionBrokers = new ArrayList<>();
        listAssignmentPartitionBrokers.add(2);
        listAssignmentPartitionBrokers.add(0);
        listAssignmentPartitionBrokers.add(1);
        List<Integer> listAssignmentPartitionBrokers1 = new ArrayList<>();
        listAssignmentPartitionBrokers1.add(1);
        listAssignmentPartitionBrokers1.add(2);
        listAssignmentPartitionBrokers1.add(0);
        List<Integer> listAssignmentPartitionBrokers2 = new ArrayList<>();
        listAssignmentPartitionBrokers2.add(0);
        listAssignmentPartitionBrokers2.add(1);
        listAssignmentPartitionBrokers2.add(2);
        List<TopicAssignment> listReassignmentsAssignment = new ArrayList<>();
        listReassignmentsAssignment.add(
            new TopicAssignment()
                .withPartition(0)
                .withPartitionBrokers(listAssignmentPartitionBrokers2)
        );
        listReassignmentsAssignment.add(
            new TopicAssignment()
                .withPartition(1)
                .withPartitionBrokers(listAssignmentPartitionBrokers1)
        );
        listReassignmentsAssignment.add(
            new TopicAssignment()
```

```
            .withPartition(2)
            .withPartitionBrokers(listAssignmentPartitionBrokers)
    );
    List<Integer> listReassignmentsBrokers = new ArrayList<>();
    listReassignmentsBrokers.add(0);
    listReassignmentsBrokers.add(1);
    listReassignmentsBrokers.add(4);
    List<Integer> listAssignmentPartitionBrokers3 = new ArrayList<>();
    listAssignmentPartitionBrokers3.add(2);
    listAssignmentPartitionBrokers3.add(0);
    listAssignmentPartitionBrokers3.add(1);
    List<Integer> listAssignmentPartitionBrokers4 = new ArrayList<>();
    listAssignmentPartitionBrokers4.add(1);
    listAssignmentPartitionBrokers4.add(2);
    listAssignmentPartitionBrokers4.add(0);
    List<Integer> listAssignmentPartitionBrokers5 = new ArrayList<>();
    listAssignmentPartitionBrokers5.add(0);
    listAssignmentPartitionBrokers5.add(1);
    listAssignmentPartitionBrokers5.add(2);
    List<TopicAssignment> listReassignmentsAssignment1 = new ArrayList<>();
    listReassignmentsAssignment1.add(
        new TopicAssignment()
            .withPartition(0)
            .withPartitionBrokers(listAssignmentPartitionBrokers5)
    );
    listReassignmentsAssignment1.add(
        new TopicAssignment()
            .withPartition(1)
            .withPartitionBrokers(listAssignmentPartitionBrokers4)
    );
    listReassignmentsAssignment1.add(
        new TopicAssignment()
            .withPartition(2)
            .withPartitionBrokers(listAssignmentPartitionBrokers3)
    );
    List<Integer> listReassignmentsBrokers1 = new ArrayList<>();
    listReassignmentsBrokers1.add(0);
    listReassignmentsBrokers1.add(1);
    listReassignmentsBrokers1.add(2);
    List<PartitionReassignEntity> listbodyReassignments = new ArrayList<>();
    listbodyReassignments.add(
        new PartitionReassignEntity()
            .withTopic("topic-1513476102")
            .withBrokers(listReassignmentsBrokers1)
            .withReplicationFactor(3)
            .withAssignment(listReassignmentsAssignment1)
    );
    listbodyReassignments.add(
        new PartitionReassignEntity()
            .withTopic("topic-1513558717")
            .withBrokers(listReassignmentsBrokers)
            .withReplicationFactor(3)
            .withAssignment(listReassignmentsAssignment)
    );
    body.withTimeEstimate(false);
    body.withThrottle(10000000);
    body.withReassignments(listbodyReassignments);
    request.withBody(body);
    try {
        CreateReassignmentTaskResponse response = client.createReassignmentTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateReassignmentTaskRequest()
        request.instance_id = "{instance_id}"
        listPartitionBrokersAssignment = [
            2,
            0,
            1
        ]
        listPartitionBrokersAssignment1 = [
            1,
            2,
            0
        ]
        listPartitionBrokersAssignment2 = [
            0,
            1,
            2
        ]
        listAssignmentReassignments = [
            TopicAssignment(
                partition=0,
                partition_brokers=listPartitionBrokersAssignment2
            ),
            TopicAssignment(
                partition=1,
                partition_brokers=listPartitionBrokersAssignment1
            ),
            TopicAssignment(
                partition=2,
                partition_brokers=listPartitionBrokersAssignment
            )
        ]
        listBrokersReassignments = [
            0,
            1,
            4
```

```
    ]
    listPartitionBrokersAssignment3 = [
        2,
        0,
        1
    ]
    listPartitionBrokersAssignment4 = [
        1,
        2,
        0
    ]
    listPartitionBrokersAssignment5 = [
        0,
        1,
        2
    ]
    listAssignmentReassignments1 = [
        TopicAssignment(
            partition=0,
            partition_brokers=listPartitionBrokersAssignment5
        ),
        TopicAssignment(
            partition=1,
            partition_brokers=listPartitionBrokersAssignment4
        ),
        TopicAssignment(
            partition=2,
            partition_brokers=listPartitionBrokersAssignment3
        )
    ]
    listBrokersReassignments1 = [
        0,
        1,
        2
    ]
    listReassignmentsbody = [
        PartitionReassignEntity(
            topic="topic-1513476102",
            brokers=listBrokersReassignments1,
            replication_factor=3,
            assignment=listAssignmentReassignments1
        ),
        PartitionReassignEntity(
            topic="topic-1513558717",
            brokers=listBrokersReassignments,
            replication_factor=3,
            assignment=listAssignmentReassignments
        )
    ]
    request.body = PartitionReassignRequest(
        time_estimate=False,
        throttle=10000000,
        reassignments=listReassignmentsbody
    )
    response = client.create_reassignment_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"
```

```
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateReassignmentTaskRequest{}
    request.InstanceId = "{instance_id}"
    var listPartitionBrokersAssignment = []int32{
        int32(2),
        int32(0),
        int32(1),
    }
    var listPartitionBrokersAssignment1 = []int32{
        int32(1),
        int32(2),
        int32(0),
    }
    var listPartitionBrokersAssignment2 = []int32{
        int32(0),
        int32(1),
        int32(2),
    }
    partitionAssignment:= int32(0)
    partitionAssignment1:= int32(1)
    partitionAssignment2:= int32(2)
    var listAssignmentReassignments = []model.TopicAssignment{
        {
            Partition: &partitionAssignment,
            PartitionBrokers: &listPartitionBrokersAssignment2,
        },
        {
            Partition: &partitionAssignment1,
            PartitionBrokers: &listPartitionBrokersAssignment1,
        },
        {
            Partition: &partitionAssignment2,
            PartitionBrokers: &listPartitionBrokersAssignment,
        },
    }
    var listBrokersReassignments = []int32{
        int32(0),
        int32(1),
        int32(4),
    }
    var listPartitionBrokersAssignment3 = []int32{
        int32(2),
        int32(0),
```

```
        int32(1),
    }
    var listPartitionBrokersAssignment4 = []int32{
        int32(1),
        int32(2),
        int32(0),
    }
    var listPartitionBrokersAssignment5 = []int32{
        int32(0),
        int32(1),
        int32(2),
    }
    partitionAssignment3:= int32(0)
    partitionAssignment4:= int32(1)
    partitionAssignment5:= int32(2)
    var listAssignmentReassignments1 = []model.TopicAssignment{
        {
            Partition: &partitionAssignment3,
            PartitionBrokers: &listPartitionBrokersAssignment5,
        },
        {
            Partition: &partitionAssignment4,
            PartitionBrokers: &listPartitionBrokersAssignment4,
        },
        {
            Partition: &partitionAssignment5,
            PartitionBrokers: &listPartitionBrokersAssignment3,
        },
    }
    var listBrokersReassignments1 = []int32{
        int32(0),
        int32(1),
        int32(2),
    }
    replicationFactorReassignments:= int32(3)
    replicationFactorReassignments1:= int32(3)
    var listReassignmentsbody = []model.PartitionReassignEntity{
        {
            Topic: "topic-1513476102",
            Brokers: &listBrokersReassignments1,
            ReplicationFactor: &replicationFactorReassignments,
            Assignment: &listAssignmentReassignments1,
        },
        {
            Topic: "topic-1513558717",
            Brokers: &listBrokersReassignments,
            ReplicationFactor: &replicationFactorReassignments1,
            Assignment: &listAssignmentReassignments,
        },
    }
    timeEstimatePartitionReassignRequest:= false
    throttlePartitionReassignRequest:= int32(10000000)
    request.Body = &model.PartitionReassignRequest{
        TimeEstimate: &timeEstimatePartitionReassignRequest,
        Throttle: &throttlePartitionReassignRequest,
        Reassignments: listReassignmentsbody,
    }
    response, err := client.CreateReassignmentTask(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	提交分区平衡任务成功（若为预估时间任务则返回预估时间）。

错误码

请参见**错误码**。

5.2.12 关闭 Kafka Manager

功能介绍

关闭Kafka Manager，相应的原来开放出的management相关接口也将不可用。

调用方法

请参见**如何调用API**。

URI

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

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

请求参数

无

响应参数

无

请求示例

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

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CloseKafkaManagerSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CloseKafkaManagerRequest request = new CloseKafkaManagerRequest();
        request.withInstanceId("{instance_id}");
        try {
            CloseKafkaManagerResponse response = client.closeKafkaManager(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

import os
```

```
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CloseKafkaManagerRequest()
        request.instance_id = "{instance_id}"
        response = client.close_kafka_manager(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CloseKafkaManagerRequest{}
    request.InstanceId = "{instance_id}"
```

```
    response, err := client.CloseKafkaManager(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	执行成功

错误码

请参见**错误码**。

5.2.13 删除用户/客户端流控配置

功能介绍

该接口用于向Kafka实例提交删除用户、客户端级别的流控任务，若成功则返回流控任务的job_id。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

表 5-71 路径参数

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

请求参数

表 5-72 请求 Body 参数

参数	是否必选	参数类型	描述
user	否	String	用户名。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
client	否	String	客户端ID。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
user-default	否	Boolean	是否使用用户默认设置。 ● 是，表示对全部用户限流。此时不能同时设置用户名。 ● 否，表示对特定用户限流。此时需要设置用户名。
client-default	否	Boolean	是否使用客户端默认设置。 ● 是，表示对全部客户端限流。此时不能设置客户端ID。 ● 否，表示对特定客户端限流。此时需要设置客户端ID。

响应参数

状态码：200

表 5-73 响应 Body 参数

参数	参数类型	描述
job_id	String	删除流控配置的任务ID

请求示例

删除用户/客户端流控配置。

```
DELETE https://{endpoint}/v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

{
  "user" : "",
  "client" : "",
  "user-default" : false,
```

```
  "client-default" : true
}
```

响应示例

状态码：200

删除用户/客户端流控配置成功。

```
{
  "job_id" : "ff8080828bdc0f64018bdcadfd8f00d7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

删除用户/客户端流控配置。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class DeleteKafkaUserClientQuotaTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteKafkaUserClientQuotaTaskRequest request = new DeleteKafkaUserClientQuotaTaskRequest();
        request.withInstanceId("{instance_id}");
        DeleteKafkaUserClientQuotaTaskReq body = new DeleteKafkaUserClientQuotaTaskReq();
        body.withClientDefault(true);
        body.withUserDefault(false);
        body.withClient("");
        body.withUser("");
        request.withBody(body);
        try {
            DeleteKafkaUserClientQuotaTaskResponse response = client.deleteKafkaUserClientQuotaTask(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

删除用户/客户端流控配置。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = DeleteKafkaUserClientQuotaTaskRequest()
        request.instance_id = "{instance_id}"
        request.body = DeleteKafkaUserClientQuotaTaskReq(
            client_default=True,
            user_default=False,
            client="",
            user=""
        )
        response = client.delete_kafka_user_client_quota_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

删除用户/客户端流控配置。

```
package main

import (
    "fmt"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteKafkaUserClientQuotaTaskRequest{}
    request.InstanceId = "{instance_id}"
    clientDefaultDeleteKafkaUserClientQuotaTaskReq:= true
    userDefaultDeleteKafkaUserClientQuotaTaskReq:= false
    clientDeleteKafkaUserClientQuotaTaskReq:= ""
    userDeleteKafkaUserClientQuotaTaskReq:= ""
    request.Body = &model.DeleteKafkaUserClientQuotaTaskReq{
        ClientDefault: &clientDefaultDeleteKafkaUserClientQuotaTaskReq,
        UserDefault: &userDefaultDeleteKafkaUserClientQuotaTaskReq,
        Client: &clientDeleteKafkaUserClientQuotaTaskReq,
        User: &userDeleteKafkaUserClientQuotaTaskReq,
    }
    response, err := client.DeleteKafkaUserClientQuotaTask(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	删除用户/客户端流控配置成功。

错误码

请参见**错误码**。

5.2.14 查询用户/客户端流控配置

功能介绍

该接口用于向Kafka实例查询流控的配置，若成功则返回流控配置的列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

表 5-74 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 5-75 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	偏移量，表示查询该偏移量后面的记录。
limit	否	Integer	查询返回记录的数量限制。

请求参数

无

响应参数

状态码：200

表 5-76 响应 Body 参数

参数	参数类型	描述
quotas	Array of **Quota** objects	客户端流控配置列表。
count	Integer	用户/客户端流控配置数目。

表 5-77 Quota

参数	参数类型	描述
user	String	用户名。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
client	String	客户端ID。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
user-default	Boolean	是否使用用户默认设置。 • 是，表示对全部用户限流。此时不能同时设置用户名。 • 否，表示对特定用户限流。此时需要设置用户名。
client-default	Boolean	是否使用客户端默认设置。 • 是，表示对全部客户端限流。此时不能设置客户端ID。 • 否，表示对特定客户端限流。此时需要设置客户端ID。
producer-byte-rate	Long	生产上限速率（单位为B/s）。
consumer-byte-rate	Long	消费上限速率（单位为B/s）。 **说明** “生产上限速率”和“消费上限速率”不可同时为空。

请求示例

无

响应示例

状态码：200

查询客户端流控配置成功。

```
{
  "quotas" : [ {
    "user" : "",
    "client" : "",
    "user-default" : false,
    "client-default" : true,
    "producer-byte-rate" : 2097152,
    "consumer-byte-rate" : 2097152
  } ],
  "count" : 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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowKafkaUserClientQuotaSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowKafkaUserClientQuotaRequest request = new ShowKafkaUserClientQuotaRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowKafkaUserClientQuotaResponse response = client.showKafkaUserClientQuota(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowKafkaUserClientQuotaRequest()
        request.instance_id = "{instance_id}"
        response = client.show_kafka_user_client_quota(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowKafkaUserClientQuotaRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowKafkaUserClientQuota(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	查询客户端流控配置成功。

错误码

请参见**错误码**。

5.2.15 创建用户/客户端流控配置

功能介绍

该接口用于向Kafka实例提交创建用户、客户端级别的流控任务，若成功则返回流控任务的job_id。

调用方法

请参见**如何调用API**。

URI

POST /v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

表 5-78 路径参数

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

请求参数

表 5-79 请求 Body 参数

参数	是否必选	参数类型	描述
user	否	String	用户名。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
client	否	String	客户端ID。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
user-default	否	Boolean	是否使用用户默认设置。 • 是，表示对全部用户限流。此时不能同时设置用户名。 • 否，表示对特定用户限流。此时需要设置用户名。
client-default	否	Boolean	是否使用客户端默认设置。 • 是，表示对全部客户端限流。此时不能设置客户端ID。 • 否，表示对特定客户端限流。此时需要设置客户端ID。
producer-byte-rate	否	Long	生产上限速率（单位为B/s）。
consumer-byte-rate	否	Long	消费上限速率（单位为B/s）。 **说明** “生产上限速率”和“消费上限速率”不可同时为空。

响应参数

状态码：200

表 5-80 响应 Body 参数

参数	参数类型	描述
job_id	String	创建流控配置的任务ID

请求示例

创建用户/客户端流控配置。

```
POST https://{endpoint}/v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

{
  "user" : "",
  "client" : "",
  "user-default" : false,
  "client-default" : true,
  "producer-byte-rate" : 3145728,
  "consumer-byte-rate" : 2097152
}
```

响应示例

状态码：200

创建用户/客户端流控配置成功。

```
{
  "job_id" : "ff8080828bdc0f64018bdcadfd8f00d7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建用户/客户端流控配置。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateKafkaUserClientQuotaTaskSolution {

    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);

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

```
        CreateKafkaUserClientQuotaTaskRequest request = new CreateKafkaUserClientQuotaTaskRequest();
        request.withInstanceId("{instance_id}");
        CreateKafkaUserClientQuotaTaskReq body = new CreateKafkaUserClientQuotaTaskReq();
        body.withConsumerByteRate(2097152L);
        body.withProducerByteRate(3145728L);
        body.withClientDefault(true);
        body.withUserDefault(false);
        body.withClient("");
        body.withUser("");
        request.withBody(body);
        try {
            CreateKafkaUserClientQuotaTaskResponse response =
client.createKafkaUserClientQuotaTask(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

创建用户/客户端流控配置。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateKafkaUserClientQuotaTaskRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateKafkaUserClientQuotaTaskReq(
            consumer_byte_rate=2097152,
            producer_byte_rate=3145728,
            client_default=True,
            user_default=False,
            client="",
            user=""
        )
        response = client.create_kafka_user_client_quota_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

创建用户/客户端流控配置。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateKafkaUserClientQuotaTaskRequest{}
    request.InstanceId = "{instance_id}"
    consumerByteRateCreateKafkaUserClientQuotaTaskReq:= int64(2097152)
    producerByteRateCreateKafkaUserClientQuotaTaskReq:= int64(3145728)
    clientDefaultCreateKafkaUserClientQuotaTaskReq:= true
    userDefaultCreateKafkaUserClientQuotaTaskReq:= false
    clientCreateKafkaUserClientQuotaTaskReq:= ""
    userCreateKafkaUserClientQuotaTaskReq:= ""
    request.Body = &model.CreateKafkaUserClientQuotaTaskReq{
        ConsumerByteRate: &consumerByteRateCreateKafkaUserClientQuotaTaskReq,
        ProducerByteRate: &producerByteRateCreateKafkaUserClientQuotaTaskReq,
        ClientDefault: &clientDefaultCreateKafkaUserClientQuotaTaskReq,
        UserDefault: &userDefaultCreateKafkaUserClientQuotaTaskReq,
        Client: &clientCreateKafkaUserClientQuotaTaskReq,
        User: &userCreateKafkaUserClientQuotaTaskReq,
    }
    response, err := client.CreateKafkaUserClientQuotaTask(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	创建用户/客户端流控配置成功。

错误码

请参见**错误码**。

5.2.16 修改用户/客户端流控配置

功能介绍

该接口用于向Kafka实例提交修改用户、客户端级别的流控任务，若成功则返回流控任务的job_id。

调用方法

请参见**如何调用API**。

URI

PUT /v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

表 5-81 路径参数

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

请求参数

表 5-82 请求 Body 参数

参数	是否必选	参数类型	描述
user	否	String	用户名。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
client	否	String	客户端ID。 不对全部用户/客户端限流时，用户名和客户端ID不能同时为空。
user-default	否	Boolean	是否使用用户默认设置。 • 是，表示对全部用户限流。此时不能同时设置用户名。 • 否，表示对特定用户限流。此时需要设置用户名。
client-default	否	Boolean	是否使用客户端默认设置。 • 是，表示对全部客户端限流。此时不能设置客户端ID。 • 否，表示对特定客户端限流。此时需要设置客户端ID。
producer-byte-rate	否	Long	生产上限速率（单位为B/s）。
consumer-byte-rate	否	Long	消费上限速率（单位为B/s）。 **说明** “生产上限速率”和“消费上限速率”不可同时为空。

响应参数

状态码：200

表 5-83 响应 Body 参数

参数	参数类型	描述
job_id	String	修改流控配置的任务ID

请求示例

修改用户/客户端流控配置。

```
PUT https://{endpoint}/v2/kafka/{project_id}/instances/{instance_id}/kafka-user-client-quota

{
  "user" : "",
  "client" : "",
  "user-default" : false,
  "client-default" : true,
  "producer-byte-rate" : 3145728,
  "consumer-byte-rate" : 2097152
}
```

响应示例

状态码：200

修改用户/客户端流配置成功。

```
{
  "job_id" : "8abfa7b38ba79a20018ba9afc550576a"
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改用户/客户端流控配置。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class UpdateKafkaUserClientQuotaTaskSolution {

    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);

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

```
        UpdateKafkaUserClientQuotaTaskRequest request = new UpdateKafkaUserClientQuotaTaskRequest();
        request.withInstanceId("{instance_id}");
        UpdateKafkaUserClientQuotaTaskReq body = new UpdateKafkaUserClientQuotaTaskReq();
        body.withConsumerByteRate(2097152L);
        body.withProducerByteRate(3145728L);
        body.withClientDefault(true);
        body.withUserDefault(false);
        body.withClient("");
        body.withUser("");
        request.withBody(body);
        try {
            UpdateKafkaUserClientQuotaTaskResponse response =
client.updateKafkaUserClientQuotaTask(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/客户端流控配置。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateKafkaUserClientQuotaTaskRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateKafkaUserClientQuotaTaskReq(
            consumer_byte_rate=2097152,
            producer_byte_rate=3145728,
            client_default=True,
            user_default=False,
            client="",
            user=""
        )
        response = client.update_kafka_user_client_quota_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

修改用户/客户端流控配置。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateKafkaUserClientQuotaTaskRequest{}
    request.InstanceId = "{instance_id}"
    consumerByteRateUpdateKafkaUserClientQuotaTaskReq:= int64(2097152)
    producerByteRateUpdateKafkaUserClientQuotaTaskReq:= int64(3145728)
    clientDefaultUpdateKafkaUserClientQuotaTaskReq:= true
    userDefaultUpdateKafkaUserClientQuotaTaskReq:= false
    clientUpdateKafkaUserClientQuotaTaskReq:= ""
    userUpdateKafkaUserClientQuotaTaskReq:= ""
    request.Body = &model.UpdateKafkaUserClientQuotaTaskReq{
        ConsumerByteRate: &consumerByteRateUpdateKafkaUserClientQuotaTaskReq,
        ProducerByteRate: &producerByteRateUpdateKafkaUserClientQuotaTaskReq,
        ClientDefault: &clientDefaultUpdateKafkaUserClientQuotaTaskReq,
        UserDefault: &userDefaultUpdateKafkaUserClientQuotaTaskReq,
        Client: &clientUpdateKafkaUserClientQuotaTaskReq,
        User: &userUpdateKafkaUserClientQuotaTaskReq,
    }
    response, err := client.UpdateKafkaUserClientQuotaTask(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	修改用户/客户端流配置成功。

错误码

请参见**错误码**。

5.3 Smart Connect

5.3.1 开启 Smart Connect（按需实例）

功能介绍

开启Smart Connect，提交创建Smart Connect节点任务。

当前通过调用API，只支持按需实例创建Smart Connect节点。

调用方法

请参见**如何调用API**。

URI

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

表 5-84 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 5-85 请求 Body 参数

参数	是否必选	参数类型	描述
specification	否	String	部署Smart Connect的规格，基准带宽，表示单位时间内传送的最大数据量。请保持和当前实例规格一致。仅老规格实例需要填写。 取值范围： ● 100MB ● 300MB ● 600MB ● 1200MB
node_cnt	否	String	Smart Connect节点数量。不能小于2个。 如果不填，默认是2个。
spec_code	否	String	转储节点规格编码。仅老规格实例需要填写。

响应参数

状态码：200

表 5-86 响应 Body 参数

参数	参数类型	描述
job_id	String	任务ID。
connector_id	String	实例转储ID。

请求示例

- 新规格按需实例的开启Smart Connect，设置Smart Connect节点数量为2。

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

{
  "node_cnt" : 2
}
```

- 老规格按需实例的开启Smart Connect，设置Smart Connect节点规格为100MB，节点数量为2。

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

{
  "specification" : "100MB",
```

```
  "node_cnt" : 2,
  "spec_code" : "kafka.c3.mini.connector"
}
```

响应示例

状态码：200

开启Smart Connect成功。

```
{
  "job_id" : "7c3ec20c-11de-4df9-acc0-7ef1dea25dfe",
  "connector_id" : "55b78880-9077-4c74-ad5a-6868555f76a4"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 新规格按需实例的开启Smart Connect，设置Smart Connect节点数量为2。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateConnectorSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConnectorRequest request = new CreateConnectorRequest();
        request.withInstanceId("{instance_id}");
        CreateConnectorReq body = new CreateConnectorReq();
        body.withNodeCnt("2");
        request.withBody(body);
        try {
            CreateConnectorResponse response = client.createConnector(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());
        }
    }
}
```

- 老规格按需实例的开启Smart Connect，设置Smart Connect节点规格为100MB，节点数量为2。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateConnectorSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConnectorRequest request = new CreateConnectorRequest();
        request.withInstanceId("{instance_id}");
        CreateConnectorReq body = new CreateConnectorReq();
        body.withSpecCode("kafka.c3.mini.connector");
        body.withNodeCnt("2");
        body.withSpecification(CreateConnectorReq.SpecificationEnum.fromValue("100MB"));
        request.withBody(body);
        try {
            CreateConnectorResponse response = client.createConnector(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

- 新规格按需实例的开启Smart Connect，设置Smart Connect节点数量为2。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateConnectorRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateConnectorReq(
            node_cnt="2"
        )
        response = client.create_connector(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)
```

- 老规格按需实例的开启Smart Connect，设置Smart Connect节点规格为100MB，节点数量为2。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

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

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

try:
    request = CreateConnectorRequest()
    request.instance_id = "{instance_id}"
    request.body = CreateConnectorReq(
        spec_code="kafka.c3.mini.connector",
        node_cnt="2",
        specification="100MB"
    )
    response = client.create_connector(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

- 新规格按需实例的开启Smart Connect，设置Smart Connect节点数量为2。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConnectorRequest{}
    request.InstanceId = "{instance_id}"
    nodeCntCreateConnectorReq:= "2"
    request.Body = &model.CreateConnectorReq{
        NodeCnt: &nodeCntCreateConnectorReq,
    }
    response, err := client.CreateConnector(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
```

```
    }
}
```

- 老规格按需实例的开启Smart Connect，设置Smart Connect节点规格为100MB，节点数量为2。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConnectorRequest{}
    request.InstanceId = "{instance_id}"
    specCodeCreateConnectorReq:= "kafka.c3.mini.connector"
    nodeCntCreateConnectorReq:= "2"
    specificationCreateConnectorReq:= model.GetCreateConnectorReqSpecificationEnum().E_100_MB
    request.Body = &model.CreateConnectorReq{
        SpecCode: &specCodeCreateConnectorReq,
        NodeCnt: &nodeCntCreateConnectorReq,
        Specification: &specificationCreateConnectorReq,
    }
    response, err := client.CreateConnector(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	开启Smart Connect成功。

错误码

请参见**错误码**。

5.3.2 关闭 Smart Connect（按需实例）

功能介绍

介绍按需实例如何关闭Smart Connect。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/kafka/instances/{instance_id}/delete-connector

表 5-87 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 5-88 响应 Body 参数

参数	参数类型	描述
job_id	String	返回异步执行删除任务的job id。

请求示例

```
POST https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/delete-connector
```

响应示例

状态码：200

提交关闭Smart Connect任务成功。

```
{
  "job_id" : "d366178c-29ea-4d5c-a344-fa2ece4a1836"
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class DeleteConnectorSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteConnectorRequest request = new DeleteConnectorRequest();
        request.withInstanceId("{instance_id}");
        try {
            DeleteConnectorResponse response = client.deleteConnector(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = DeleteConnectorRequest()
        request.instance_id = "{instance_id}"
        response = client.delete_connector(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteConnectorRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.DeleteConnector(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	提交关闭Smart Connect任务成功。

错误码

请参见**错误码**。

5.3.3 创建 Smart Connect 任务

功能介绍

创建Smart Connect任务。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/connector/tasks

表 5-89 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 5-90 请求 Body 参数

参数	是否必选	参数类型	描述
task_name	否	String	SmartConnect任务名称。
start_later	否	Boolean	是否稍后再启动任务。如需要创建任务后立即启动，请填false；如希望稍后在任务列表中手动开启任务，请填true。
topics	否	String	SmartConnect任务配置的Topic。
topics_regex	否	String	SmartConnect任务配置的Topic正则表达式。
source_type	否	String	SmartConnect任务的源端类型。
source_task	否	**SmartConnectTaskReqSourceConfig** object	SmartConnect任务的源端配置。
sink_type	否	String	SmartConnect任务的目标端类型。
sink_task	否	**SmartConnectTaskReqSinkConfig** object	SmartConnect任务的目标端配置。

表 5-91 SmartConnectTaskReqSourceConfig

参数	是否必选	参数类型	描述
redis_address	否	String	Redis实例地址。（仅源端类型为Redis时需要填写）
redis_type	否	String	Redis实例类型。（仅源端类型为Redis时需要填写）
dcs_instance_id	否	String	DCS实例ID。（仅源端类型为Redis时需要填写）
redis_password	否	String	Redis密码。（仅源端类型为Redis时需要填写）

参数	是否必选	参数类型	描述
sync_mode	否	String	同步类型，“RDB_ONLY”为全量同步，“CUSTOM_OFFSET”为全量同步+增量同步。（仅源端类型为Redis时需要填写）
full_sync_wait_ms	否	Integer	全量同步重试间隔时间，单位：毫秒。（仅源端类型为Redis时需要填写）
full_sync_max_retry	否	Integer	全量同步最大重试次数。（仅源端类型为Redis时需要填写）
ratelimit	否	Integer	限速，单位为KB/s。-1表示不限速。（仅源端类型为Redis时需要填写）
current_cluster_name	否	String	当前Kafka实例别名。（仅源端类型为Kafka时需要填写）
cluster_name	否	String	对端Kafka实例别名。（仅源端类型为Kafka时需要填写）
user_name	否	String	对端Kafka开启SASL_SSL时设置的用户名，或者创建SASL_SSL用户时设置的用户名。（仅源端类型为Kafka且对端Kafka认证方式为“SASL_SSL”时需要填写）
password	否	String	对端Kafka开启SASL_SSL时设置的密码，或者创建SASL_SSL用户时设置的密码。（仅源端类型为Kafka且对端Kafka认证方式为“SASL_SSL”时需要填写）
sasl_mechanism	否	String	对端Kafka认证机制。（仅源端类型为Kafka且“认证方式”为“SASL_SSL”时需要填写）
instance_id	否	String	对端Kafka实例ID。（仅源端类型为Kafka时需要填写，instance_id和bootstrap_servers仅需要填写其中一个）
bootstrap_servers	否	String	对端Kafka实例地址。（仅源端类型为Kafka时需要填写，instance_id和bootstrap_servers仅需要填写其中一个）

参数	是否必选	参数类型	描述
security_protocol	否	String	对端Kafka认证方式。（仅源端类型为Kafka需要填写） 支持以下两种认证方式： ● SASL_SSL：表示实例已开启SASL_SSL。 ● PLAINTEXT：表示实例未开启SASL_SSL。
direction	否	String	同步方向；pull为把对端Kafka实例数据复制到当前Kafka实例中，push为把当前Kafka实例数据复制到对端Kafka实例中，two-way为对两端Kafka实例数据进行双向复制。（仅源端类型为Kafka时需要填写）
sync_consumer_offsets_enabled	否	Boolean	是否同步消费进度。（仅源端类型为Kafka时需要填写）
replication_factor	否	Integer	在对端实例中自动创建Topic时，指定Topic的副本数，此参数值不能超过对端实例的代理数。如果对端实例中设置了“default.replication.factor”，此参数的优先级高于“default.replication.factor”。（仅源端类型为Kafka时需要填写）
task_num	否	Integer	数据复制的任务数。默认值为2，建议保持默认值。如果“同步方式”为“双向”，实际任务数=设置的任务数*2。（仅源端类型为Kafka时需要填写）
rename_topic_enabled	否	Boolean	是否重命名Topic，在目标Topic名称前添加源端Kafka实例的别名，形成目标Topic新的名称。（仅源端类型为Kafka时需要填写）
provenance_header_enabled	否	Boolean	目标Topic接收复制的消息，此消息header中包含消息来源。两端实例数据双向复制时，请开启“添加来源header”，防止循环复制。（仅源端类型为Kafka时需要填写）

参数	是否必选	参数类型	描述
consumer_strategy	否	String	启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅源端类型为Kafka时需要填写）
compression_type	否	String	复制消息所使用的压缩算法。（仅源端类型为Kafka时需要填写） • none • gzip • snappy • lz4 • zstd
topics_mapping	否	String	topic映射，用于自定义目标端Topic名称。不能同时设置“重命名Topic”和“topic映射”。topic映射请按照“源端topic:目的端topic”的格式填写，如涉及多个topic映射，请用“,”分隔开，例如：topic-sc-1:topic-sc-2,topic-sc-3:topic-sc-4。（仅源端类型为Kafka时需要填写）

表 5-92 SmartConnectTaskReqSinkConfig

参数	是否必选	参数类型	描述
redis_address	否	String	Redis实例地址。（仅目标端类型为Redis时需要填写）
redis_type	否	String	Redis实例类型。（仅目标端类型为Redis时需要填写）
dcs_instance_id	否	String	DCS实例ID。（仅目标端类型为Redis时需要填写）
redis_password	否	String	Redis密码。（仅目标端类型为Redis时需要填写）
consumer_strategy	否	String	转储启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅目标端类型为OBS时需要填写）
destination_file_type	否	String	转储文件格式。当前只支持TEXT。（仅目标端类型为OBS时需要填写）

参数	是否必选	参数类型	描述
deliver_time_interval	否	Integer	数据转储周期（秒），默认配置为300秒。（仅目标端类型为OBS时需要填写）
access_key	否	String	AK，访问密钥ID。（仅目标端类型为OBS时需要填写）
secret_key	否	String	SK，与访问密钥ID结合使用的密钥。（仅目标端类型为OBS时需要填写）
obs_bucket_name	否	String	转储地址，即存储Topic数据的OBS桶的名称。（仅目标端类型为OBS时需要填写）
obs_path	否	String	转储目录，即OBS中存储Topic的目录，多级目录可以用“/”进行分隔。（仅目标端类型为OBS时需要填写）
partition_format	否	String	时间目录格式。（仅目标端类型为OBS时需要填写） • yyyy：年 • yyyy/MM：年/月 • yyyy/MM/dd：年/月/日 • yyyy/MM/dd/HH：年/月/日/时 • yyyy/MM/dd/HH/mm：年/月/日/时/分
record_delimiter	否	String	记录分行符，用于分隔写入转储文件的用户数据。（仅目标端类型为OBS时需要填写） 取值范围： • 逗号“,” • 分号“;” • 竖线“|” • 换行符“\n” • NULL
store_keys	否	Boolean	是否转储Key，开启表示转储Key，关闭表示不转储Key。（仅目标端类型为OBS时需要填写）

响应参数

状态码：200

表 5-93 响应 Body 参数

参数	参数类型	描述
task_name	String	SmartConnect任务名称。
topics	String	SmartConnect任务配置的Topic。
topics_regex	String	SmartConnect任务配置的Topic正则表达式。
source_type	String	SmartConnect任务的源端类型。
source_task	**SmartConnectTaskRespSourceConfig** object	SmartConnect任务的源端配置。
sink_type	String	SmartConnect任务的目标端类型。
sink_task	**SmartConnectTaskRespSinkConfig** object	SmartConnect任务的目标端配置。
id	String	SmartConnect任务的id。
status	String	SmartConnect任务的状态。
create_time	Long	SmartConnect任务的创建时间。

表 5-94 SmartConnectTaskRespSourceConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅源端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅源端类型为Redis时会显示）
dcs_instance_id	String	DCS实例ID。（仅源端类型为Redis时会显示）
sync_mode	String	同步类型，“RDB_ONLY”为全量同步，“CUSTOM_OFFSET”为全量同步+增量同步。（仅源端类型为Redis时会显示）
full_sync_wait_ms	Integer	全量同步重试间隔时间，单位：毫秒。（仅源端类型为Redis时会显示）
full_sync_max_retry	Integer	全量同步最大重试次数。（仅源端类型为Redis时会显示）
ratelimit	Integer	限速，单位为KB/s。-1表示不限速（仅源端类型为Redis时会显示）

参数	参数类型	描述
current_cluster_name	String	当前Kafka实例别名。（仅源端类型为Kafka时会显示）
cluster_name	String	对端Kafka实例别名。（仅源端类型为Kafka时会显示）
user_name	String	对端Kafka用户名。（仅源端类型为Kafka时会显示）
sasl_mechanism	String	对端Kafka认证机制。（仅源端类型为Kafka时会显示）
instance_id	String	对端Kafka实例ID。（仅源端类型为Kafka时会显示）
bootstrap_servers	String	对端Kafka实例地址。（仅源端类型为Kafka时会显示）
security_protocol	String	对端Kafka认证方式。（仅源端类型为Kafka时会显示）
direction	String	同步方向。（仅源端类型为Kafka时会显示）
sync_consumer_offsets_enabled	Boolean	是否同步消费进度。（仅源端类型为Kafka时会显示）
replication_factor	Integer	副本数。（仅源端类型为Kafka时会显示）
task_num	Integer	任务数。（仅源端类型为Kafka时会显示）
rename_topic_enabled	Boolean	是否重命名Topic。（仅源端类型为Kafka时会显示）
provenance_header_enabled	Boolean	是否添加来源header。（仅源端类型为Kafka时会显示）
consumer_strategy	String	启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅源端类型为Kafka时会显示）
compression_type	String	压缩算法。（仅源端类型为Kafka时会显示）
topics_mapping	String	topic映射。（仅源端类型为Kafka时会显示）

表 5-95 SmartConnectTaskRespSinkConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅目标端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅目标端类型为Redis时会显示）
dcs_instance_id	String	DCS实例ID。（仅目标端类型为Redis时会显示）
target_db	Integer	目标数据库，默认为-1。（仅目标端类型为Redis时会显示）
consumer_strategy	String	转储启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅目标端类型为OBS时会显示）
destination_file_type	String	转储文件格式。当前只支持TEXT。（仅目标端类型为OBS时会显示）
deliver_time_interval	Integer	记数据转储周期（秒）。（仅目标端类型为OBS时会显示）
obs_bucket_name	String	转储地址。（仅目标端类型为OBS时会显示）
obs_path	String	转储目录。（仅目标端类型为OBS时会显示）
partition_format	String	时间目录格式。（仅目标端类型为OBS时会显示）
record_delimiter	String	记录分行符。（仅目标端类型为OBS时会显示）
store_keys	Boolean	存储Key。（仅目标端类型为OBS时会显示）
obs_part_size	Integer	每个传输文件多大后就开始上传，单位为byte；默认值5242880。（仅目标端类型为OBS时会显示）
flush_size	Integer	flush_size。（仅目标端类型为OBS时会显示）
timezone	String	时区。（仅目标端类型为OBS时会显示）
schema_generator_class	String	schema_generator类，默认为"io.confluent.connect.storage.hive.schema.DefaultSchemaGenerator"。（仅目标端类型为OBS时会显示）

参数	参数类型	描述
partitioner_class	String	partitioner类，默认"io.confluent.connect.storage.partitioner.TimeBasedPartitioner"。（仅目标端类型为OBS时会显示）
value_converter	String	value_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
key_converter	String	key_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
kv_delimiter	String	kv_delimiter，默认为":"。（仅目标端类型为OBS时会显示）

请求示例

- 创建一个立即启动的转储任务。

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

{
  "task_name" : "smart-connect-1",
  "start_later" : false,
  "source_type" : "NONE",
  "topics_regex" : "topic-obs*",
  "sink_type" : "OBS_SINK",
  "sink_task" : {
    "consumer_strategy" : "earliest",
    "destination_file_type" : "TEXT",
    "deliver_time_interval" : 300,
    "access_key" : "********",
    "secret_key" : "********",
    "obs_bucket_name" : "obs_bucket",
    "obs_path" : "obsTransfer-1810125534",
    "partition_format" : "yyyy/MM/dd/HH/mm",
    "record_delimiter" : "\\n",
    "store_keys" : false
  }
}
```

- 创建一个稍后启动的Kafka数据复制任务。

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

{
  "task_name" : "smart-connect-2",
  "start_later" : true,
  "source_type" : "KAFKA_REPLICATOR_SOURCE",
  "source_task" : {
    "current_cluster_name" : "A",
    "cluster_name" : "B",
    "user_name" : "user1",
    "password" : "********",
    "sasl_mechanism" : "SCRAM-SHA-512",
    "instance_id" : "b54c9dd8-********-********",
    "direction" : "two-way",
    "sync_consumer_offsets_enabled" : false,
```

```
  "replication_factor" : 3,
  "task_num" : 2,
  "rename_topic_enabled" : false,
  "provenance_header_enabled" : true,
  "consumer_strategy" : "latest",
  "compression_type" : "snappy",
  "topics_mapping" : "topic-sc-1:topic-sc-3,topic-sc-2:topic-sc-4"
 }
}
```

- 创建一个立即启动的Redis数据复制任务，同步方式为全量同步，配置全量同步最大重试次数为10，重试间隔时间为10000毫秒，带宽限制为10KB/s。

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

{
 "task_name" : "smart-connect-3",
 "start_later" : false,
 "source_type" : "REDIS_REPLICATOR_SOURCE",
 "source_task" : {
  "redis_address" : "192.168.99.249:6379,192.168.120.127:6379,192.168.116.219:6379",
  "redis_type" : "cluster",
  "redis_password" : "********",
  "sync_mode" : "RDB_ONLY",
  "full_sync_max_retry" : 10,
  "full_sync_wait_ms" : 10000,
  "ratelimit" : 10
 },
 "topics" : "topic-sc-3",
 "sink_type" : "REDIS_REPLICATOR_SINK",
 "sink_task" : {
  "redis_address" : "192.168.119.51:6379",
  "redis_type" : "standalone",
  "redis_password" : "********"
 }
}
```

响应示例

状态码：200

创建Smart Connect任务成功。

```
{
 "task_name" : "smart-connect-121248117",
 "topics" : "topic-sc",
 "source_task" : {
  "redis_address" : "192.168.91.179:6379",
  "redis_type" : "standalone",
  "dcs_instance_id" : "949190a2-598a-4afd-99a8-dad3cae1e7cd",
  "sync_mode" : "RDB_ONLY,",
  "full_sync_wait_ms" : 13000,
  "full_sync_max_retry" : 4,
  "ratelimit" : -1
 },
 "source_type" : "REDIS_REPLICATOR_SOURCE",
 "sink_task" : {
  "redis_address" : "192.168.119.51:6379",
  "redis_type" : "standalone",
  "dcs_instance_id" : "9b981368-a8e3-416a-87d9-1581a968b41b",
  "target_db" : -1
 },
 "sink_type" : "REDIS_REPLICATOR_SINK",
 "id" : "8a205bbd-7181-4b5e-9bd6-37274ce84577",
 "status" : "RUNNING",
 "create_time" : 1708427753133
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 创建一个立即启动的转储任务。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConnectorTaskRequest request = new CreateConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        CreateSmartConnectTaskReq body = new CreateSmartConnectTaskReq();
        SmartConnectTaskReqSinkConfig sinkTaskbody = new SmartConnectTaskReqSinkConfig();
        sinkTaskbody.withConsumerStrategy("earliest")
            .withDestinationFileType("TEXT")
            .withDeliverTimeInterval(300)
            .withAccessKey("********")
            .withSecretKey("********")
            .withObsBucketName("obs_bucket")
            .withObsPath("obsTransfer-1810125534")
            .withPartitionFormat("yyyy/MM/dd/HH/mm")
            .withRecordDelimiter("\n")
            .withStoreKeys(false);
        body.withSinkTask(sinkTaskbody);
        body.withSinkType(CreateSmartConnectTaskReq.SinkTypeEnum.fromValue("OBS_SINK"));
        body.withSourceType(CreateSmartConnectTaskReq.SourceTypeEnum.fromValue("NONE"));
        body.withTopicsRegex("topic-obs*");
        body.withStartLater(false);
        body.withTaskName("smart-connect-1");
        request.withBody(body);
        try {
            CreateConnectorTaskResponse response = client.createConnectorTask(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());
        }
    }
}
```

- 创建一个稍后启动的Kafka数据复制任务。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConnectorTaskRequest request = new CreateConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        CreateSmartConnectTaskReq body = new CreateSmartConnectTaskReq();
        SmartConnectTaskReqSourceConfig sourceTaskbody = new SmartConnectTaskReqSourceConfig();
        sourceTaskbody.withCurrentClusterName("A")
            .withClusterName("B")
            .withUserName("user1")
            .withPassword("********")
            .withSaslMechanism("SCRAM-SHA-512")
            .withInstanceId("b54c9dd8-********-********")
            .withDirection("two-way")
            .withSyncConsumerOffsetsEnabled(false)
            .withReplicationFactor(3)
            .withTaskNum(2)
            .withRenameTopicEnabled(false)
            .withProvenanceHeaderEnabled(true)
            .withConsumerStrategy("latest")
            .withCompressionType("snappy")
            .withTopicsMapping("topic-sc-1:topic-sc-3,topic-sc-2:topic-sc-4");
        body.withSourceTask(sourceTaskbody);

body.withSourceType(CreateSmartConnectTaskReq.SourceTypeEnum.fromValue("KAFKA_REPLICATOR_SOURCE"));
        body.withStartLater(true);
```

```
        body.withTaskName("smart-connect-2");
        request.withBody(body);
        try {
            CreateConnectorTaskResponse response = client.createConnectorTask(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());
        }
    }
}
```

- 创建一个立即启动的Redis数据复制任务，同步方式为全量同步，配置全量同步最大重试次数为10，重试间隔时间为10000毫秒，带宽限制为10KB/s。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateConnectorTaskRequest request = new CreateConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        CreateSmartConnectTaskReq body = new CreateSmartConnectTaskReq();
        SmartConnectTaskReqSinkConfig sinkTaskbody = new SmartConnectTaskReqSinkConfig();
        sinkTaskbody.withRedisAddress("192.168.119.51:6379")
            .withRedisType("standalone")
            .withRedisPassword("********");
        SmartConnectTaskReqSourceConfig sourceTaskbody = new SmartConnectTaskReqSourceConfig();

sourceTaskbody.withRedisAddress("192.168.99.249:6379,192.168.120.127:6379,192.168.116.219:6379")
            .withRedisType("cluster")
            .withRedisPassword("********")
            .withSyncMode("RDB_ONLY")
            .withFullSyncWaitMs(10000)
```

```
            .withFullSyncMaxRetry(10)
            .withRatelimit(10);
        body.withSinkTask(sinkTaskbody);

body.withSinkType(CreateSmartConnectTaskReq.SinkTypeEnum.fromValue("REDIS_REPLICATOR_SINK"
));
        body.withSourceTask(sourceTaskbody);

body.withSourceType(CreateSmartConnectTaskReq.SourceTypeEnum.fromValue("REDIS_REPLICATOR_S
OURCE"));
        body.withTopics("topic-sc-3");
        body.withStartLater(false);
        body.withTaskName("smart-connect-3");
        request.withBody(body);
        try {
            CreateConnectorTaskResponse response = client.createConnectorTask(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

- 创建一个立即启动的转储任务。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        sinkTaskbody = SmartConnectTaskReqSinkConfig(
            consumer_strategy="earliest",
            destination_file_type="TEXT",
            deliver_time_interval=300,
            access_key="********",
            secret_key="********",
```

```
            obs_bucket_name="obs_bucket",
            obs_path="obsTransfer-1810125534",
            partition_format="yyyy/MM/dd/HH/mm",
            record_delimiter="\n",
            store_keys=False
        )
        request.body = CreateSmartConnectTaskReq(
            sink_task=sinkTaskbody,
            sink_type="OBS_SINK",
            source_type="NONE",
            topics_regex="topic-obs*",
            start_later=False,
            task_name="smart-connect-1"
        )
        response = client.create_connector_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)
```

- 创建一个稍后启动的Kafka数据复制任务。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        sourceTaskbody = SmartConnectTaskReqSourceConfig(
            current_cluster_name="A",
            cluster_name="B",
            user_name="user1",
            password="********",
            sasl_mechanism="SCRAM-SHA-512",
            instance_id="b54c9dd8-********-********",
            direction="two-way",
            sync_consumer_offsets_enabled=False,
            replication_factor=3,
            task_num=2,
            rename_topic_enabled=False,
            provenance_header_enabled=True,
            consumer_strategy="latest",
            compression_type="snappy",
            topics_mapping="topic-sc-1:topic-sc-3,topic-sc-2:topic-sc-4"
        )
        request.body = CreateSmartConnectTaskReq(
```

```
            source_task=sourceTaskbody,
            source_type="KAFKA_REPLICATOR_SOURCE",
            start_later=True,
            task_name="smart-connect-2"
        )
        response = client.create_connector_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)
```

- 创建一个立即启动的Redis数据复制任务，同步方式为全量同步，配置全量同步最大重试次数为10，重试间隔时间为10000毫秒，带宽限制为10KB/s。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        sinkTaskbody = SmartConnectTaskReqSinkConfig(
            redis_address="192.168.119.51:6379",
            redis_type="standalone",
            redis_password="********"
        )
        sourceTaskbody = SmartConnectTaskReqSourceConfig(
            redis_address="192.168.99.249:6379,192.168.120.127:6379,192.168.116.219:6379",
            redis_type="cluster",
            redis_password="********",
            sync_mode="RDB_ONLY",
            full_sync_wait_ms=10000,
            full_sync_max_retry=10,
            ratelimit=10
        )
        request.body = CreateSmartConnectTaskReq(
            sink_task=sinkTaskbody,
            sink_type="REDIS_REPLICATOR_SINK",
            source_task=sourceTaskbody,
            source_type="REDIS_REPLICATOR_SOURCE",
            topics="topic-sc-3",
            start_later=False,
            task_name="smart-connect-3"
        )
        response = client.create_connector_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

- 创建一个立即启动的转储任务。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    consumerStrategySinkTask:= "earliest"
    destinationFileTypeSinkTask:= "TEXT"
    deliverTimeIntervalSinkTask:= int32(300)
    accessKeySinkTask:= "********"
    secretKeySinkTask:= "********"
    obsBucketNameSinkTask:= "obs_bucket"
    obsPathSinkTask:= "obsTransfer-1810125534"
    partitionFormatSinkTask:= "yyyy/MM/dd/HH/mm"
    recordDelimiterSinkTask:= "\n"
    storeKeysSinkTask:= false
    sinkTaskbody := &model.SmartConnectTaskReqSinkConfig{
        ConsumerStrategy: &consumerStrategySinkTask,
        DestinationFileType: &destinationFileTypeSinkTask,
        DeliverTimeInterval: &deliverTimeIntervalSinkTask,
        AccessKey: &accessKeySinkTask,
        SecretKey: &secretKeySinkTask,
        ObsBucketName: &obsBucketNameSinkTask,
        ObsPath: &obsPathSinkTask,
        PartitionFormat: &partitionFormatSinkTask,
        RecordDelimiter: &recordDelimiterSinkTask,
        StoreKeys: &storeKeysSinkTask,
    }
    sinkTypeCreateSmartConnectTaskReq:=
model.GetCreateSmartConnectTaskReqSinkTypeEnum().OBS_SINK
    sourceTypeCreateSmartConnectTaskReq:=
```

```
model.GetCreateSmartConnectTaskReqSourceTypeEnum().NONE
    topicsRegexCreateSmartConnectTaskReq:= "topic-obs*"
    startLaterCreateSmartConnectTaskReq:= false
    taskNameCreateSmartConnectTaskReq:= "smart-connect-1"
    request.Body = &model.CreateSmartConnectTaskReq{
        SinkTask: sinkTaskbody,
        SinkType: &sinkTypeCreateSmartConnectTaskReq,
        SourceType: &sourceTypeCreateSmartConnectTaskReq,
        TopicsRegex: &topicsRegexCreateSmartConnectTaskReq,
        StartLater: &startLaterCreateSmartConnectTaskReq,
        TaskName: &taskNameCreateSmartConnectTaskReq,
    }
    response, err := client.CreateConnectorTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建一个稍后启动的Kafka数据复制任务。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    currentClusterNameSourceTask:= "A"
    clusterNameSourceTask:= "B"
    userNameSourceTask:= "user1"
    passwordSourceTask:= "********"
    saslMechanismSourceTask:= "SCRAM-SHA-512"
    instanceIdSourceTask:= "b54c9dd8-********-********"
    directionSourceTask:= "two-way"
    syncConsumerOffsetsEnabledSourceTask:= false
    replicationFactorSourceTask:= int32(3)
    taskNumSourceTask:= int32(2)
    renameTopicEnabledSourceTask:= false
    provenanceHeaderEnabledSourceTask:= true
    consumerStrategySourceTask:= "latest"
    compressionTypeSourceTask:= "snappy"
```

```
    topicsMappingSourceTask:= "topic-sc-1:topic-sc-3,topic-sc-2:topic-sc-4"
    sourceTaskbody := &model.SmartConnectTaskReqSourceConfig{
        CurrentClusterName: &currentClusterNameSourceTask,
        ClusterName: &clusterNameSourceTask,
        UserName: &userNameSourceTask,
        Password: &passwordSourceTask,
        SaslMechanism: &saslMechanismSourceTask,
        InstanceId: &instanceIdSourceTask,
        Direction: &directionSourceTask,
        SyncConsumerOffsetsEnabled: &syncConsumerOffsetsEnabledSourceTask,
        ReplicationFactor: &replicationFactorSourceTask,
        TaskNum: &taskNumSourceTask,
        RenameTopicEnabled: &renameTopicEnabledSourceTask,
        ProvenanceHeaderEnabled: &provenanceHeaderEnabledSourceTask,
        ConsumerStrategy: &consumerStrategySourceTask,
        CompressionType: &compressionTypeSourceTask,
        TopicsMapping: &topicsMappingSourceTask,
    }
    sourceTypeCreateSmartConnectTaskReq:=
model.GetCreateSmartConnectTaskReqSourceTypeEnum().KAFKA_REPLICATOR_SOURCE
    startLaterCreateSmartConnectTaskReq:= true
    taskNameCreateSmartConnectTaskReq:= "smart-connect-2"
    request.Body = &model.CreateSmartConnectTaskReq{
        SourceTask: sourceTaskbody,
        SourceType: &sourceTypeCreateSmartConnectTaskReq,
        StartLater: &startLaterCreateSmartConnectTaskReq,
        TaskName: &taskNameCreateSmartConnectTaskReq,
    }
    response, err := client.CreateConnectorTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 创建一个立即启动的Redis数据复制任务，同步方式为全量同步，配置全量同步最大重试次数为10，重试间隔时间为10000毫秒，带宽限制为10KB/s。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
```

```
        Build())

    request := &model.CreateConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    redisAddressSinkTask:= "192.168.119.51:6379"
    redisTypeSinkTask:= "standalone"
    redisPasswordSinkTask:= "********"
    sinkTaskbody := &model.SmartConnectTaskReqSinkConfig{
        RedisAddress: &redisAddressSinkTask,
        RedisType: &redisTypeSinkTask,
        RedisPassword: &redisPasswordSinkTask,
    }
    redisAddressSourceTask:= "192.168.99.249:6379,192.168.120.127:6379,192.168.116.219:6379"
    redisTypeSourceTask:= "cluster"
    redisPasswordSourceTask:= "********"
    syncModeSourceTask:= "RDB_ONLY"
    fullSyncWaitMsSourceTask:= int32(10000)
    fullSyncMaxRetrySourceTask:= int32(10)
    ratelimitSourceTask:= int32(10)
    sourceTaskbody := &model.SmartConnectTaskReqSourceConfig{
        RedisAddress: &redisAddressSourceTask,
        RedisType: &redisTypeSourceTask,
        RedisPassword: &redisPasswordSourceTask,
        SyncMode: &syncModeSourceTask,
        FullSyncWaitMs: &fullSyncWaitMsSourceTask,
        FullSyncMaxRetry: &fullSyncMaxRetrySourceTask,
        Ratelimit: &ratelimitSourceTask,
    }
    sinkTypeCreateSmartConnectTaskReq:=
model.GetCreateSmartConnectTaskReqSinkTypeEnum().REDIS_REPLICATOR_SINK
    sourceTypeCreateSmartConnectTaskReq:=
model.GetCreateSmartConnectTaskReqSourceTypeEnum().REDIS_REPLICATOR_SOURCE
    topicsCreateSmartConnectTaskReq:= "topic-sc-3"
    startLaterCreateSmartConnectTaskReq:= false
    taskNameCreateSmartConnectTaskReq:= "smart-connect-3"
    request.Body = &model.CreateSmartConnectTaskReq{
        SinkTask: sinkTaskbody,
        SinkType: &sinkTypeCreateSmartConnectTaskReq,
        SourceTask: sourceTaskbody,
        SourceType: &sourceTypeCreateSmartConnectTaskReq,
        Topics: &topicsCreateSmartConnectTaskReq,
        StartLater: &startLaterCreateSmartConnectTaskReq,
        TaskName: &taskNameCreateSmartConnectTaskReq,
    }
    response, err := client.CreateConnectorTask(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Smart Connect任务成功。

错误码

请参见**错误码**。

5.3.4 查询 Smart Connect 任务列表

功能介绍

查询Smart Connect任务列表。

调用方法

请参见**如何调用API**。

URI

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

表 5-96 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 5-97 Query 参数

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。
limit	否	Integer	当次查询返回的最大实例个数，默认值为10，取值范围为1~50。

请求参数

无

响应参数

状态码：200

表 5-98 响应 Body 参数

参数	参数类型	描述
tasks	Array of **SmartConnectTaskEntity** objects	Smart Connector任务详情。
total_number	Integer	Smart Connector任务数。
max_tasks	Integer	Smart Connector最大任务数。
quota_tasks	Integer	Smart Connector任务配额。

表 5-99 SmartConnectTaskEntity

参数	参数类型	描述
task_name	String	SmartConnect任务名称。
topics	String	SmartConnect任务配置的Topic。
topics_regex	String	SmartConnect任务配置的Topic正则表达式。
source_type	String	SmartConnect任务的源端类型。
source_task	**SmartConnectTaskRespSourceConfig** object	SmartConnect任务的源端配置。
sink_type	String	SmartConnect任务的目标端类型。
sink_task	**SmartConnectTaskRespSinkConfig** object	SmartConnect任务的目标端配置。
id	String	SmartConnect任务的id。
status	String	SmartConnect任务的状态。
create_time	Long	SmartConnect任务的创建时间。

表 5-100 SmartConnectTaskRespSourceConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅源端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅源端类型为Redis时会显示）

参数	参数类型	描述
dcs_instance_id	String	DCS实例ID。（仅源端类型为Redis时会显示）
sync_mode	String	同步类型，“RDB_ONLY”为全量同步，“CUSTOM_OFFSET”为全量同步+增量同步。（仅源端类型为Redis时会显示）
full_sync_wait_ms	Integer	全量同步重试间隔时间，单位：毫秒。（仅源端类型为Redis时会显示）
full_sync_max_retry	Integer	全量同步最大重试次数。（仅源端类型为Redis时会显示）
ratelimit	Integer	限速，单位为KB/s。-1表示不限速（仅源端类型为Redis时会显示）
current_cluster_name	String	当前Kafka实例别名。（仅源端类型为Kafka时会显示）
cluster_name	String	对端Kafka实例别名。（仅源端类型为Kafka时会显示）
user_name	String	对端Kafka用户名。（仅源端类型为Kafka时会显示）
sasl_mechanism	String	对端Kafka认证机制。（仅源端类型为Kafka时会显示）
instance_id	String	对端Kafka实例ID。（仅源端类型为Kafka时会显示）
bootstrap_servers	String	对端Kafka实例地址。（仅源端类型为Kafka时会显示）
security_protocol	String	对端Kafka认证方式。（仅源端类型为Kafka时会显示）
direction	String	同步方向。（仅源端类型为Kafka时会显示）
sync_consumer_offsets_enabled	Boolean	是否同步消费进度。（仅源端类型为Kafka时会显示）
replication_factor	Integer	副本数。（仅源端类型为Kafka时会显示）
task_num	Integer	任务数。（仅源端类型为Kafka时会显示）
rename_topic_enabled	Boolean	是否重命名Topic。（仅源端类型为Kafka时会显示）
provenance_header_enabled	Boolean	是否添加来源header。（仅源端类型为Kafka时会显示）

参数	参数类型	描述
consumer_strategy	String	启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅源端类型为Kafka时会显示）
compression_type	String	压缩算法。（仅源端类型为Kafka时会显示）
topics_mapping	String	topic映射。（仅源端类型为Kafka时会显示）

表 5-101 SmartConnectTaskRespSinkConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅目标端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅目标端类型为Redis时会显示）
dcs_instance_id	String	DCS实例ID。（仅目标端类型为Redis时会显示）
target_db	Integer	目标数据库，默认为-1。（仅目标端类型为Redis时会显示）
consumer_strategy	String	转储启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅目标端类型为OBS时会显示）
destination_file_type	String	转储文件格式。当前只支持TEXT。（仅目标端类型为OBS时会显示）
deliver_time_interval	Integer	记数据转储周期（秒）。（仅目标端类型为OBS时会显示）
obs_bucket_name	String	转储地址。（仅目标端类型为OBS时会显示）
obs_path	String	转储目录。（仅目标端类型为OBS时会显示）
partition_format	String	时间目录格式。（仅目标端类型为OBS时会显示）
record_delimiter	String	记录分行符。（仅目标端类型为OBS时会显示）
store_keys	Boolean	存储Key。（仅目标端类型为OBS时会显示）

参数	参数类型	描述
obs_part_size	Integer	每个传输文件多大后就开始上传，单位为byte；默认值5242880。（仅目标端类型为OBS时会显示）
flush_size	Integer	flush_size。（仅目标端类型为OBS时会显示）
timezone	String	时区。（仅目标端类型为OBS时会显示）
schema_generator_class	String	schema_generator类，默认为"io.confluent.connect.storage.hive.schema.DefaultSchemaGenerator"。（仅目标端类型为OBS时会显示）
partitioner_class	String	partitioner类，默认"io.confluent.connect.storage.partitioner.TimeBasedPartitioner"。（仅目标端类型为OBS时会显示）
value_converter	String	value_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
key_converter	String	key_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
kv_delimiter	String	kv_delimiter，默认为":"。（仅目标端类型为OBS时会显示）

请求示例

无

响应示例

状态码：200

查询Smart Connect任务列表成功。

```
{
  "tasks" : [ {
    "task_name" : "smart-connect-1571576841",
    "topics" : "topic-1643449744",
    "source_task" : {
      "current_cluster_name" : "A",
      "cluster_name" : "B",
      "direction" : "pull",
      "bootstrap_servers" : "192.168.45.58:9092,192.168.44.1:9092,192.168.41.230:9092,192.168.43.112:9092",
      "instance_id" : "59f6d088-****-****-****-********",
      "consumer_strategy" : "earliest",
      "sync_consumer_offsets_enabled" : false,
```

```
    "rename_topic_enabled" : false,
    "provenance_header_enabled" : false,
    "security_protocol" : "PLAINTEXT",
    "sasl_mechanism" : "PLAIN",
    "user_name" : "",
    "topics_mapping" : "",
    "compression_type" : "none",
    "task_num" : 2,
    "replication_factor" : 3
  },
  "source_type" : "KAFKA_REPLICATOR_SOURCE",
  "sink_task" : null,
  "sink_type" : "NONE",
  "id" : "194917d0-****-****-****-********",
  "status" : "RUNNING",
  "create_time" : 1708427753133
}, {
  "task_name" : "smart-connect-1",
  "topics_regex" : "topic-obs*",
  "source_task" : null,
  "source_type" : "NONE",
  "sink_task" : {
    "consumer_strategy" : "earliest",
    "destination_file_type" : "TEXT",
    "obs_bucket_name" : "abcabc",
    "obs_path" : "obsTransfer-1810125534",
    "partition_format" : "yyyy/MM/dd/HH/mm",
    "record_delimiter" : "\\n",
    "deliver_time_interva" : 300,
    "obs_part_size" : 5242880,
    "flush_size" : 1000000,
    "timezone" : "Asia/Chongqing",
    "schema_generator_class" : "io.confluent.connect.storage.hive.schema.DefaultSchemaGenerator",
    "partitioner_class" : "io.confluent.connect.storage.partitioner.TimeBasedPartitioner",
    "value_converter" : "org.apache.kafka.connect.converters.ByteArrayConverter",
    "key_converter" : "org.apache.kafka.connect.converters.ByteArrayConverter",
    "store_keys" : false,
    "kv_delimiter" : ":"
  },
  "sink_type" : "OBS_SINK",
  "id" : "3c0ac4d1-****-****-****-********",
  "status" : "RUNNING",
  "create_time" : 1708565483911
}, {
  "task_name" : "smart-connect-121248117",
  "topics" : "topic-sc",
  "source_task" : {
    "redis_address" : "192.168.91.179:6379",
    "redis_type" : "standalone",
    "dcs_instance_id" : "949190a2-598a-4afd-99a8-dad3cae1e7cd",
    "sync_mode" : "RDB_ONLY",
    "full_sync_wait_ms" : 13000,
    "full_sync_max_retry" : 4,
    "ratelimit" : -1
  },
  "source_type" : "REDIS_REPLICATOR_SOURCE",
  "sink_task" : {
    "redis_address" : "192.168.119.51:6379",
    "redis_type" : "standalone",
    "dcs_instance_id" : "9b981368-a8e3-416a-87d9-1581a968b41b",
    "target_db" : -1
  },
  "sink_type" : "REDIS_REPLICATOR_SINK",
  "id" : "8a205bbd-****-****-****-********",
  "status" : "RUNNING",
  "create_time" : 1708427753133
} ],
"total_number" : 3,
"max_tasks" : 18,
```

```
  "quota_tasks" : 18
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListConnectorTasksSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListConnectorTasksRequest request = new ListConnectorTasksRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListConnectorTasksResponse response = client.listConnectorTasks(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
```

```
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListConnectorTasksRequest()
        request.instance_id = "{instance_id}"
        response = client.list_connector_tasks(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListConnectorTasksRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListConnectorTasks(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Smart Connect任务列表成功。

错误码

请参见**错误码**。

5.3.5 查询 Smart Connector 任务详情

功能介绍

查询Smart Connector任务详情。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}

表 5-102 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。
task_id	是	String	Smart Connector任务ID。

请求参数

无

响应参数

状态码：200

表 5-103 响应 Body 参数

参数	参数类型	描述
task_name	String	SmartConnect任务名称。
topics	String	SmartConnect任务配置的Topic。
topics_regex	String	SmartConnect任务配置的Topic正则表达式。
source_type	String	SmartConnect任务的源端类型。
source_task	**SmartConnectTaskRespSourceConfig** object	SmartConnect任务的源端配置。
sink_type	String	SmartConnect任务的目标端类型。
sink_task	**SmartConnectTaskRespSinkConfig** object	SmartConnect任务的目标端配置。
id	String	SmartConnect任务的id。
status	String	SmartConnect任务的状态。
create_time	Long	SmartConnect任务的创建时间。

表 5-104 SmartConnectTaskRespSourceConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅源端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅源端类型为Redis时会显示）
dcs_instance_id	String	DCS实例ID。（仅源端类型为Redis时会显示）
sync_mode	String	同步类型，“RDB_ONLY”为全量同步，“CUSTOM_OFFSET”为全量同步+增量同步。（仅源端类型为Redis时会显示）
full_sync_wait_ms	Integer	全量同步重试间隔时间，单位：毫秒。（仅源端类型为Redis时会显示）
full_sync_max_retry	Integer	全量同步最大重试次数。（仅源端类型为Redis时会显示）

参数	参数类型	描述
ratelimit	Integer	限速，单位为KB/s。-1表示不限速（仅源端类型为Redis时会显示）
current_cluster_name	String	当前Kafka实例别名。（仅源端类型为Kafka时会显示）
cluster_name	String	对端Kafka实例别名。（仅源端类型为Kafka时会显示）
user_name	String	对端Kafka用户名。（仅源端类型为Kafka时会显示）
sasl_mechanism	String	对端Kafka认证机制。（仅源端类型为Kafka时会显示）
instance_id	String	对端Kafka实例ID。（仅源端类型为Kafka时会显示）
bootstrap_servers	String	对端Kafka实例地址。（仅源端类型为Kafka时会显示）
security_protocol	String	对端Kafka认证方式。（仅源端类型为Kafka时会显示）
direction	String	同步方向。（仅源端类型为Kafka时会显示）
sync_consumer_offsets_enabled	Boolean	是否同步消费进度。（仅源端类型为Kafka时会显示）
replication_factor	Integer	副本数。（仅源端类型为Kafka时会显示）
task_num	Integer	任务数。（仅源端类型为Kafka时会显示）
rename_topic_enabled	Boolean	是否重命名Topic。（仅源端类型为Kafka时会显示）
provenance_header_enabled	Boolean	是否添加来源header。（仅源端类型为Kafka时会显示）
consumer_strategy	String	启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅源端类型为Kafka时会显示）
compression_type	String	压缩算法。（仅源端类型为Kafka时会显示）
topics_mapping	String	topic映射。（仅源端类型为Kafka时会显示）

表 5-105 SmartConnectTaskRespSinkConfig

参数	参数类型	描述
redis_address	String	Redis实例地址。（仅目标端类型为Redis时会显示）
redis_type	String	Redis实例类型。（仅目标端类型为Redis时会显示）
dcs_instance_id	String	DCS实例ID。（仅目标端类型为Redis时会显示）
target_db	Integer	目标数据库，默认为-1。（仅目标端类型为Redis时会显示）
consumer_strategy	String	转储启动偏移量，latest为获取最新的数据，earliest为获取最早的数据。（仅目标端类型为OBS时会显示）
destination_file_type	String	转储文件格式。当前只支持TEXT。（仅目标端类型为OBS时会显示）
deliver_time_interval	Integer	记数据转储周期（秒）。（仅目标端类型为OBS时会显示）
obs_bucket_name	String	转储地址。（仅目标端类型为OBS时会显示）
obs_path	String	转储目录。（仅目标端类型为OBS时会显示）
partition_format	String	时间目录格式。（仅目标端类型为OBS时会显示）
record_delimiter	String	记录分行符。（仅目标端类型为OBS时会显示）
store_keys	Boolean	存储Key。（仅目标端类型为OBS时会显示）
obs_part_size	Integer	每个传输文件多大后就开始上传，单位为byte；默认值5242880。（仅目标端类型为OBS时会显示）
flush_size	Integer	flush_size。（仅目标端类型为OBS时会显示）
timezone	String	时区。（仅目标端类型为OBS时会显示）
schema_generator_class	String	schema_generator类，默认为"io.confluent.connect.storage.hive.schema.DefaultSchemaGenerator"。（仅目标端类型为OBS时会显示）

参数	参数类型	描述
partitioner_class	String	partitioner类，默认"io.confluent.connect.storage.partitioner.TimeBasedPartitioner"。（仅目标端类型为OBS时会显示）
value_converter	String	value_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
key_converter	String	key_converter，默认为"org.apache.kafka.connect.converters.ByteArrayConverter"。（仅目标端类型为OBS时会显示）
kv_delimiter	String	kv_delimiter，默认为":"。（仅目标端类型为OBS时会显示）

请求示例

无

响应示例

状态码：200

查询Smart Connector任务详情成功。

```
{
  "task_name" : "smart-connect-121248117",
  "topics" : "topic-sc",
  "source_task" : {
    "redis_address" : "192.168.91.179:6379",
    "redis_type" : "standalone",
    "dcs_instance_id" : "949190a2-598a-4afd-99a8-dad3cae1e7cd",
    "sync_mode" : "RDB_ONLY,",
    "full_sync_wait_ms" : 13000,
    "full_sync_max_retry" : 4,
    "ratelimit" : -1
  },
  "source_type" : "REDIS_REPLICATOR_SOURCE",
  "sink_task" : {
    "redis_address" : "192.168.119.51:6379",
    "redis_type" : "standalone",
    "dcs_instance_id" : "9b981368-a8e3-416a-87d9-1581a968b41b",
    "target_db" : -1
  },
  "sink_type" : "REDIS_REPLICATOR_SINK",
  "id" : "8a205bbd-7181-4b5e-9bd6-37274ce84577",
  "status" : "RUNNING",
  "create_time" : 1708427753133
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowConnectorTaskRequest request = new ShowConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            ShowConnectorTaskResponse response = client.showConnectorTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.show_connector_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.ShowConnectorTask(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Smart Connector任务详情成功。

错误码

请参见**错误码**。

5.3.6 删除 Smart Connector 任务

功能介绍

删除Smart Connector任务。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}

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。
task_id	是	String	Smart Connector任务ID。

请求参数

无

响应参数

无

请求示例

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/connector/tasks/{task_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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class DeleteConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteConnectorTaskRequest request = new DeleteConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            DeleteConnectorTaskResponse response = client.deleteConnectorTask(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
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = DeleteConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.delete_connector_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())
```

```
    request := &model.DeleteConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.DeleteConnectorTask(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Smart Connector任务成功。

错误码

请参见**错误码**。

5.3.7 暂停 Smart Connect 任务

功能介绍

暂停Smart Connect任务。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/pause

表 5-107 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。
task_id	是	String	Smart Connect任务ID。

请求参数

无

响应参数

无

请求示例

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/pause
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class PauseConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        PauseConnectorTaskRequest request = new PauseConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            PauseConnectorTaskResponse response = client.pauseConnectorTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = PauseConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.pause_connector_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.PauseConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.PauseConnectorTask(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	暂停Smart Connect任务成功。

错误码

请参见**错误码**。

5.3.8 启动已暂停的 Smart Connect 任务

功能介绍

启动已暂停的Smart Connect任务。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/resume

表 5-108 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。
task_id	是	String	Smart Connect任务ID。

请求参数

无

响应参数

无

请求示例

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/resume
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResumeConnectorTaskSolution {

    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);

    KafkaClient client = KafkaClient.newBuilder()
            .withCredential(auth)
            .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
            .build();
    ResumeConnectorTaskRequest request = new ResumeConnectorTaskRequest();
    request.withInstanceId("{instance_id}");
    request.withTaskId("{task_id}");
    try {
        ResumeConnectorTaskResponse response = client.resumeConnectorTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResumeConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.resume_connector_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResumeConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.ResumeConnectorTask(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	启动已暂停的Smart Connect任务成功。

错误码

请参见**错误码**。

5.3.9 启动未启动的 Smart Connect 任务/重启已暂停或者运行中的 Smart Connect 任务

功能介绍

用于**启动未启动的Smart Connect任务**以及**重启已暂停或者运行中的Smart Connect任务**。注意，重启Smart Connect任务将重置同步进度，并重新开始同步任务。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/kafka/instances/{instance_id}/connector/tasks/{task_id}/restart

表 5-109 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。
task_id	是	String	Smart Connect任务ID。

请求参数

无

响应参数

无

请求示例

```
PUT https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/connector/tasks/{task_id}/restart
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class RestartConnectorTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        RestartConnectorTaskRequest request = new RestartConnectorTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            RestartConnectorTaskResponse response = client.restartConnectorTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
```

```
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = RestartConnectorTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.restart_connector_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.RestartConnectorTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.RestartConnectorTask(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	启动未启动的Smart Connect任务/重启已暂停或者运行中的Smart Connect任务成功。

错误码

请参见**错误码**。

5.4 规格变更管理

5.4.1 实例扩容

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扩容。

调用方法

请参见**如何调用API**。

URI

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

表 5-110 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。
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 5-111 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	扩容后的存储空间。 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。
new_broker_num	否	Integer	当oper_type参数为horizontal时，该参数有效。 取值范围：最多支持30个broker。
new_product_id	否	String	垂直扩容时的新产品ID。 当oper_type类型是vertical时，该参数才有效且必填。 产品ID可以从**查询产品规格列表**获取。
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类型是horizontal时，该参数必填。

参数	是否必选	参数类型	描述
tenant_ips	否	Array of strings	指定的内网IP地址，仅支持指定IPv4。 指定的IP数量只能小于等于新增节点数量。 当指定IP小于节点数量时，未指定的节点随机分配内网IP地址。
second_tenant_subnet_id	否	String	实例扩容时新节点使用备用子网的id。 当实例扩容使用备用子网，则传入此值。 需要联系客服添加白名单才能传入此值。

响应参数

状态码：200

表 5-112 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

- 扩容存储空间（按需实例）。

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

{
  "oper_type" : "storage",
  "new_storage_space" : 600
}
```

- 扩容代理数量（按需实例）。

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

{
  "oper_type" : "horizontal",
  "new_storage_space" : 1600,
  "new_broker_num" : 4,
  "tenant_ips" : [ "127.xx.xx.x", "127.xx.xx.x", "127.xx.xx.x" ]
}
```

- 扩容代理规格（按需实例）。

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

{
  "oper_type" : "vertical",
  "new_product_id" : "c6.4u8g.cluster"
}
```

响应示例

状态码：200

实例扩容成功。

```
{
  "job_id" : "93b94287-728d-4bb1-a158-cb66cb0854e7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

- 扩容存储空间（按需实例）。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResizeEngineInstanceSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        body.withNewStorageSpace(600);
        body.withOperType("storage");
        request.withBody(body);
        try {
            ResizeEngineInstanceResponse response = client.resizeEngineInstance(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());
        }
    }
}
```

- 扩容代理数量（按需实例）。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class ResizeEngineInstanceSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        List<String> listbodyTenantIps = new ArrayList<>();
        listbodyTenantIps.add("127.xx.xx.x");
        listbodyTenantIps.add("127.xx.xx.x");
        listbodyTenantIps.add("127.xx.xx.x");
        body.withTenantIps(listbodyTenantIps);
        body.withNewBrokerNum(4);
        body.withNewStorageSpace(1600);
        body.withOperType("horizontal");
        request.withBody(body);
        try {
            ResizeEngineInstanceResponse response = client.resizeEngineInstance(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());
        }
    }
}
```

- 扩容代理规格（按需实例）。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResizeEngineInstanceSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        request.withInstanceId("{instance_id}");
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        body.withNewProductId("c6.4u8g.cluster");
        body.withOperType("vertical");
        request.withBody(body);
        try {
            ResizeEngineInstanceResponse response = client.resizeEngineInstance(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

- 扩容存储空间（按需实例）。

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

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResizeEngineInstanceRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.body = ResizeEngineInstanceReq(
            new_storage_space=600,
            oper_type="storage"
        )
        response = client.resize_engine_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)
```

- 扩容代理数量（按需实例）。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResizeEngineInstanceRequest()
        request.engine = "{engine}"
```

```
        request.instance_id = "{instance_id}"
        listTenantIpsbody = [
            "127.xx.xx.x",
            "127.xx.xx.x",
            "127.xx.xx.x"
        ]
        request.body = ResizeEngineInstanceReq(
            tenant_ips=listTenantIpsbody,
            new_broker_num=4,
            new_storage_space=1600,
            oper_type="horizontal"
        )
        response = client.resize_engine_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)
```

- 扩容代理规格（按需实例）。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResizeEngineInstanceRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.body = ResizeEngineInstanceReq(
            new_product_id="c6.4u8g.cluster",
            oper_type="vertical"
        )
        response = client.resize_engine_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

- 扩容存储空间（按需实例）。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResizeEngineInstanceRequest{}
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    newStorageSpaceResizeEngineInstanceReq:= int32(600)
    request.Body = &model.ResizeEngineInstanceReq{
        NewStorageSpace: &newStorageSpaceResizeEngineInstanceReq,
        OperType: "storage",
    }
    response, err := client.ResizeEngineInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 扩容代理数量（按需实例）。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResizeEngineInstanceRequest{}
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    var listTenantIpsbody = []string{
        "127.xx.xx.x",
        "127.xx.xx.x",
        "127.xx.xx.x",
    }
    newBrokerNumResizeEngineInstanceReq:= int32(4)
    newStorageSpaceResizeEngineInstanceReq:= int32(1600)
    request.Body = &model.ResizeEngineInstanceReq{
        TenantIps: &listTenantIpsbody,
        NewBrokerNum: &newBrokerNumResizeEngineInstanceReq,
        NewStorageSpace: &newStorageSpaceResizeEngineInstanceReq,
        OperType: "horizontal",
    }
    response, err := client.ResizeEngineInstance(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 扩容代理规格（按需实例）。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

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

```
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    request.InstanceId = "{instance_id}"
    newProductIdResizeEngineInstanceReq:= "c6.4u8g.cluster"
    request.Body = &model.ResizeEngineInstanceReq{
        NewProductId: &newProductIdResizeEngineInstanceReq,
        OperType: "vertical",
    }
    response, err := client.ResizeEngineInstance(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	实例扩容成功。

错误码

请参见**错误码**。

5.4.2 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 5-113 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎。
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 5-114 Query 参数

参数	是否必选	参数类型	描述
type	是	String	产品的类型。 • advanced: 专享版

请求参数

无

响应参数

状态码：200

表 5-115 响应 Body 参数

参数	参数类型	描述
engine	String	消息引擎类型:kafka。
versions	Array of strings	消息引擎支持的版本。
products	Array of **ExtendProductInfoEntity** objects	规格变更的产品信息。

表 5-116 ExtendProductInfoEntity

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。
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 **ExtendProductIosEntity** objects	磁盘IO信息。
support_features	Array of **ExtendProductSupportFeaturesEntity** objects	支持的特性功能。

参数	参数类型	描述
properties	**ExtendProductPropertiesEntity** object	产品规格属性描述。
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。

表 5-117 ExtendProductIosEntity

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。
unavailable_zones	Array of strings	资源售罄的可用区列表。

表 5-118 ExtendProductSupportFeaturesEntity

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。

表 5-119 ExtendProductPropertiesEntity

参数	参数类型	描述
max_partition_per_broker	String	每个Broker的最大分区数。
max_broker	String	Broker的最大个数。
max_storage_per_node	String	每个节点的最大存储。单位为GB。
max_consumer_per_broker	String	每个Broker的最大消费者数。
min_broker	String	Broker的最小个数。
max_bandwidth_per_broker	String	每个Broker的最大带宽。

参数	参数类型	描述
min_storage_per_node	String	每个节点的最小存储。单位为GB。
max_tps_per_broker	String	每个Broker的最大TPS。
product_alias	String	product_id的别名。

请求示例

查询实例的扩容规格列表。

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/extend?type={type}
```

响应示例

状态码：200

查询实例的扩容规格列表成功。

```
{
  "engine" : "kafka",
  "versions" : [ "1.1.0", "2.7" ],
  "products" : [ {
    "type" : "cluster",
    "product_id" : "c6.2u4g.cluster",
    "ecs_flavor_id" : "c3.large.2",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
        "max_node" : "10",
        "min_task" : "1",
        "min_node" : "2"
      }
    } ],
    "properties" : {
      "max_partition_per_broker" : "250",
      "max_broker" : "30",
      "max_storage_per_node" : "10000",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "3",
      "product_alias" : "kafka.2u4g.cluster",
      "max_bandwidth_per_broker" : "100",
      "min_storage_per_node" : "100",
      "max_tps_per_broker" : "30000"
    },
    "available_zones" : [ "xxx" ],
```

```
    "unavailable_zones" : [ ]
  }, {
    "type" : "cluster",
    "product_id" : "c6.2u4g.cluster.dec",
    "ecs_flavor_id" : "c6.large.2",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.dss.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.dss.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    }, {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
        "max_node" : "10",
        "min_task" : "1",
        "min_node" : "2"
      }
    } ],
    "properties" : {
      "max_partition_per_broker" : "100",
      "max_broker" : "30",
      "max_storage_per_node" : "400",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "3",
      "product_alias" : "kafka.2u4g.cluster.dec",
      "max_bandwidth_per_broker" : "100",
      "min_storage_per_node" : "100",
      "max_tps_per_broker" : "30000"
    },
    "available_zones" : [ ],
    "unavailable_zones" : [ "xxx" ]
  }, {
    "type" : "cluster",
    "product_id" : "c6.4u8g.cluster",
    "ecs_flavor_id" : "c3.xlarge.2",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "available_zones" : [ "xxx" ],
      "type" : "evs",
      "unavailable_zones" : [ ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
```

```
    "properties" : {
      "max_task" : "10",
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "500",
    "max_broker" : "30",
    "max_storage_per_node" : "20000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "product_alias" : "kafka.4u8g.cluster",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "100",
    "max_tps_per_broker" : "100000"
  },
  "available_zones" : [ "xxx" ],
  "unavailable_zones" : [ ]
}, {
  "type" : "cluster",
  "product_id" : "c6.8u16g.cluster",
  "ecs_flavor_id" : "c3.2xlarge.2",
  "arch_types" : [ "X86" ],
  "charging_mode" : [ "monthly", "hourly" ],
  "ios" : [ {
    "io_spec" : "dms.physical.storage.high.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
    "unavailable_zones" : [ ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
    "unavailable_zones" : [ ]
  } ],
  "support_features" : [ {
    "name" : "connector_obs",
    "properties" : {
      "max_task" : "10",
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "1000",
    "max_broker" : "30",
    "max_storage_per_node" : "30000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "product_alias" : "kafka.8u16g.cluster",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "100",
    "max_tps_per_broker" : "150000"
  },
  "available_zones" : [ "xxx" ],
  "unavailable_zones" : [ ]
}, {
  "type" : "cluster",
  "product_id" : "c6.12u24g.cluster",
  "ecs_flavor_id" : "c3.3xlarge.2",
  "arch_types" : [ "X86" ],
  "charging_mode" : [ "monthly", "hourly" ],
  "ios" : [ {
    "io_spec" : "dms.physical.storage.high.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
```

```
    "unavailable_zones" : [ ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
    "unavailable_zones" : [ ]
  } ],
  "support_features" : [ {
    "name" : "connector_obs",
    "properties" : {
      "max_task" : "10",
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "1500",
    "max_broker" : "30",
    "max_storage_per_node" : "30000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "product_alias" : "kafka.12u24g.cluster",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "100",
    "max_tps_per_broker" : "200000"
  },
  "available_zones" : [ "xxx" ],
  "unavailable_zones" : [ ]
}, {
  "type" : "cluster",
  "product_id" : "c6.16u32g.cluster",
  "ecs_flavor_id" : "c3.4xlarge.2",
  "arch_types" : [ "X86" ],
  "charging_mode" : [ "monthly", "hourly" ],
  "ios" : [ {
    "io_spec" : "dms.physical.storage.high.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
    "unavailable_zones" : [ ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "available_zones" : [ "xxx" ],
    "type" : "evs",
    "unavailable_zones" : [ ]
  } ],
  "support_features" : [ {
    "name" : "connector_obs",
    "properties" : {
      "max_task" : "10",
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "2000",
    "max_broker" : "30",
    "max_storage_per_node" : "30000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "product_alias" : "kafka.16u32g.cluster",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "100",
    "max_tps_per_broker" : "250000"
  },
  "available_zones" : [ "xxx" ],
  "unavailable_zones" : [ ]
```

```
 } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowEngineInstanceExtendProductInfoRequest request = new ShowEngineInstanceExtendProductInfoRequest();

request.withEngine(ShowEngineInstanceExtendProductInfoRequest.EngineEnum.fromValue("{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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
```

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

    request := &model.ShowEngineInstanceExtendProductInfoRequest{}
    request.Engine = model.GetShowEngineInstanceExtendProductInfoRequestEngineEnum().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**。

5.5 主题管理

5.5.1 Kafka 生产消息

功能介绍

在控制台发送指定消息到Kafka实例

调用方法

请参见**如何调用API**。

URI

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

表 5-120 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

表 5-121 Query 参数

参数	是否必选	参数类型	描述
action_id	是	String	动作ID，生产消息对应的action_id为send。

请求参数

表 5-122 请求 Body 参数

参数	是否必选	参数类型	描述
topic	是	String	Kafka的topic
body	是	String	消息内容
property_list	是	Array of **property_list** objects	topic的分区信息等

表 5-123 property_list

参数	是否必选	参数类型	描述
name	否	String	特性名字
value	否	String	特性值

响应参数

状态码：200

表 5-124 响应 Body 参数

参数	参数类型	描述
topic	String	Kafka的topic
body	String	消息内容
property_list	Array of objects	topic的分区信息等

请求示例

Kafka控制台发送消息

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/messages/action?action_id={action_id}

{
  "topic" : "XXXX",
  "body" : "hello world",
  "property_list" : [ {
    "name" : "KEY",
    "value" : "testKey"
  }, {
    "name" : "PARTITION",
    "value" : "0"
  } ]
}
```

响应示例

状态码：200

生产消息成功

```
{
  "topic" : "XXXX",
  "body" : "XXXX",
  "property_list" : [ ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

Kafka控制台发送消息

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class SendKafkaMessageSolution {

    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);

    KafkaClient client = KafkaClient.newBuilder()
            .withCredential(auth)
            .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
            .build();
    SendKafkaMessageRequest request = new SendKafkaMessageRequest();
    request.withInstanceId("{instance_id}");
    SendKafkaMessageRequestBody body = new SendKafkaMessageRequestBody();
    List<SendKafkaMessageRequestBodyPropertyList> listbodyPropertyList = new ArrayList<>();
    listbodyPropertyList.add(
        new SendKafkaMessageRequestBodyPropertyList()
            .withName("KEY")
            .withValue("testKey")
    );
    listbodyPropertyList.add(
        new SendKafkaMessageRequestBodyPropertyList()
            .withName("PARTITION")
            .withValue("0")
    );
    body.withPropertyList(listbodyPropertyList);
    body.withBody("hello world");
    body.withTopic("XXXX");
    request.withBody(body);
    try {
        SendKafkaMessageResponse response = client.sendKafkaMessage(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

Kafka控制台发送消息

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
```

```
        .build()

    try:
        request = SendKafkaMessageRequest()
        request.instance_id = "{instance_id}"
        listPropertyListbody = [
            SendKafkaMessageRequestBodyPropertyList(
                name="KEY",
                value="testKey"
            ),
            SendKafkaMessageRequestBodyPropertyList(
                name="PARTITION",
                value="0"
            )
        ]
        request.body = SendKafkaMessageRequestBody(
            property_list=listPropertyListbody,
            body="hello world",
            topic="XXXX"
        )
        response = client.send_kafka_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

Kafka控制台发送消息

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.SendKafkaMessageRequest{}
    request.InstanceId = "{instance_id}"
    namePropertyList:= "KEY"
    valuePropertyList:= "testKey"
    namePropertyList1:= "PARTITION"
```

```
    valuePropertyList1:= "0"
    var listPropertyListbody = []model.SendKafkaMessageRequestBodyPropertyList{
        {
            Name: &namePropertyList,
            Value: &valuePropertyList,
        },
        {
            Name: &namePropertyList1,
            Value: &valuePropertyList1,
        },
    }
    request.Body = &model.SendKafkaMessageRequestBody{
        PropertyList: listPropertyListbody,
        Body: "hello world",
        Topic: "XXXX",
    }
    response, err := client.SendKafkaMessage(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	生产消息成功

错误码

请参见**错误码**。

5.5.2 Kafka 实例创建 Topic

功能介绍

该接口用于向Kafka实例创建Topic。

调用方法

请参见**如何调用API**。

URI

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

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。

请求参数

表 5-126 请求 Body 参数

参数	是否必选	参数类型	描述
id	是	String	topic名称，长度为3-200，以字母开头且只支持大小写字母、中横线、下划线、点以及数字。
replication	否	Integer	副本数，配置数据的可靠性。 取值范围：1-3。
sync_message_flush	否	Boolean	是否使用同步落盘。默认值为false。同步落盘会导致性能降低。
partition	否	Integer	topic分区数，设置消费的并发数。取值范围：1-200。
sync_replication	否	Boolean	是否开启同步复制，开启后，客户端生产消息时相应的也要设置acks=-1，否则不生效，默认关闭。
retention_time	否	Integer	消息老化时间。默认值为72。取值范围1-720，单位小时。
topic_other_configs	否	Array of **topic_other_configs** objects	topic配置
topic_desc	否	String	topic描述

表 5-127 topic_other_configs

参数	是否必选	参数类型	描述
name	否	String	配置名称
value	否	String	配置值

响应参数

状态码：200

表 5-128 响应 Body 参数

参数	参数类型	描述
name	String	topic名称。

请求示例

创建一个Topic，Topic名称为test01。

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

{
  "id" : "test01",
  "partition" : 3,
  "replication" : 3,
  "retention_time" : 72,
  "sync_message_flush" : false,
  "sync_replication" : "false",
  "topic_other_configs" : [ {
    "name" : "message.timestamp.type",
    "value" : "LogAppendTime"
  }, {
    "name" : "max.message.bytes",
    "value" : 10485760
  } ],
  "topic_desc" : ""
}
```

响应示例

状态码：200

创建成功，返回topic名称

```
{
  "name" : "test01"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个Topic，Topic名称为test01。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
```

```
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class CreateInstanceTopicSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateInstanceTopicRequest request = new CreateInstanceTopicRequest();
        request.withInstanceId("{instance_id}");
        CreateInstanceTopicReq body = new CreateInstanceTopicReq();
        List<CreateInstanceTopicReqTopicOtherConfigs> listbodyTopicOtherConfigs = new ArrayList<>();
        listbodyTopicOtherConfigs.add(
            new CreateInstanceTopicReqTopicOtherConfigs()
                .withName("message.timestamp.type")
                .withValue("LogAppendTime")
        );
        listbodyTopicOtherConfigs.add(
            new CreateInstanceTopicReqTopicOtherConfigs()
                .withName("max.message.bytes")
                .withValue("10485760")
        );
        body.withTopicDesc("");
        body.withTopicOtherConfigs(listbodyTopicOtherConfigs);
        body.withRetentionTime(72);
        body.withSyncReplication(false);
        body.withPartition(3);
        body.withSyncMessageFlush(false);
        body.withReplication(3);
        body.withId("test01");
        request.withBody(body);
        try {
            CreateInstanceTopicResponse response = client.createInstanceTopic(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Topic，Topic名称为test01。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateInstanceTopicRequest()
        request.instance_id = "{instance_id}"
        listTopicOtherConfigsbody = [
            CreateInstanceTopicReqTopicOtherConfigs(
                name="message.timestamp.type",
                value="LogAppendTime"
            ),
            CreateInstanceTopicReqTopicOtherConfigs(
                name="max.message.bytes",
                value="10485760"
            )
        ]
        request.body = CreateInstanceTopicReq(
            topic_desc="",
            topic_other_configs=listTopicOtherConfigsbody,
            retention_time=72,
            sync_replication=False,
            partition=3,
            sync_message_flush=False,
            replication=3,
            id="test01"
        )
        response = client.create_instance_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

创建一个Topic，Topic名称为test01。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateInstanceTopicRequest{}
    request.InstanceId = "{instance_id}"
    nameTopicOtherConfigs:= "message.timestamp.type"
    valueTopicOtherConfigs:= "LogAppendTime"
    nameTopicOtherConfigs1:= "max.message.bytes"
    valueTopicOtherConfigs1:= "10485760"
    var listTopicOtherConfigsbody = []model.CreateInstanceTopicReqTopicOtherConfigs{
        {
            Name: &nameTopicOtherConfigs,
            Value: &valueTopicOtherConfigs,
        },
        {
            Name: &nameTopicOtherConfigs1,
            Value: &valueTopicOtherConfigs1,
        },
    }
    topicDescCreateInstanceTopicReq:= ""
    retentionTimeCreateInstanceTopicReq:= int32(72)
    syncReplicationCreateInstanceTopicReq:= false
    partitionCreateInstanceTopicReq:= int32(3)
    syncMessageFlushCreateInstanceTopicReq:= false
    replicationCreateInstanceTopicReq:= int32(3)
    request.Body = &model.CreateInstanceTopicReq{
        TopicDesc: &topicDescCreateInstanceTopicReq,
        TopicOtherConfigs: &listTopicOtherConfigsbody,
        RetentionTime: &retentionTimeCreateInstanceTopicReq,
        SyncReplication: &syncReplicationCreateInstanceTopicReq,
        Partition: &partitionCreateInstanceTopicReq,
        SyncMessageFlush: &syncMessageFlushCreateInstanceTopicReq,
        Replication: &replicationCreateInstanceTopicReq,
        Id: "test01",
    }
    response, err := client.CreateInstanceTopic(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，返回topic名称

错误码

请参见**错误码**。

5.5.3 Kafka 实例查询 Topic

功能介绍

该接口用于查询指定Kafka实例的Topic详情。

调用方法

请参见**如何调用API**。

URI

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

表 5-129 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 5-130 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	当次查询返回的最大实例个数，默认值为10，取值范围为1~50。

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
total	Integer	topic总数。
size	Integer	分页查询的大小。
remain_partitions	Integer	剩余分区数。
max_partitions	Integer	分区总数。
topic_max_partitions	Integer	单个topic最大占用分区数。
topics	Array of **TopicEntity** objects	topic列表。

表 5-132 TopicEntity

参数	参数类型	描述
policiesOnly	Boolean	是否为默认策略。
name	String	topic名称。
replication	Integer	副本数，配置数据的可靠性。
partition	Integer	topic分区数，设置消费的并发数。
retention_time	Integer	消息老化时间。
sync_replication	Boolean	是否开启同步复制，开启后，客户端生产消息时相应的也要设置acks=-1，否则不生效，默认关闭。
sync_message_flush	Boolean	是否使用同步落盘。默认值为false。同步落盘会导致性能降低。
external_configs	Object	扩展配置。
topic_type	Integer	topic类型(0:普通Topic 1:系统(内部)Topic)。
topic_other_configs	Array of **topic_other_configs** objects	topic其他配置
topic_desc	String	topic描述
created_at	Long	topic创建时间

表 5-133 topic_other_configs

参数	参数类型	描述
name	String	配置名称
valid_values	String	配置有效值
default_value	String	配置默认值
config_type	String	配置类型：dynamic/static
value	String	配置值
value_type	String	配置值类型

请求示例

查询Topic列表。

```
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" : 1,
  "size" : 0,
  "topics" : [ {
    "policiesOnly" : false,
    "name" : "Topic-test01",
    "replication" : 3,
    "partition" : 3,
    "retention_time" : 72,
    "sync_replication" : "false",
    "sync_message_flush" : "false",
    "topic_other_configs" : [ {
      "name" : "max.message.bytes",
      "valid_values" : "[0...10485760]",
      "default_value" : "10485760",
      "config_type" : "dynamic",
      "value" : "10485760",
      "value_type" : "int"
    }, {
      "name" : "message.timestamp.type",
      "valid_values" : "[CreateTime, LogAppendTime]",
      "default_value" : "LogAppendTime",
      "config_type" : "dynamic",
      "value" : "LogAppendTime",
      "value_type" : "string"
    } ],
    "external_configs" : { },
    "topic_type" : 0,
    "topic_desc" : "This is a test topic",
    "created_at" : 1688112779916
  } ],
  "remain_partitions" : 294,
  "max_partitions" : 300,
  "topic_max_partitions" : 200
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListInstanceTopicsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListInstanceTopicsRequest request = new ListInstanceTopicsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListInstanceTopicsResponse response = client.listInstanceTopics(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListInstanceTopicsRequest()
        request.instance_id = "{instance_id}"
        response = client.list_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListInstanceTopicsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListInstanceTopics(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**。

5.5.4 修改 Kafka 实例 Topic

功能介绍

修改Kafka实例Topic

调用方法

请参见**如何调用API**。

URI

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

表 5-134 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 5-135 请求 Body 参数

参数	是否必选	参数类型	描述
topics	否	Array of **topics** objects	修改的topic列表。

表 5-136 topics

参数	是否必选	参数类型	描述
id	是	String	topic名称，不支持修改。
retention_time	否	Integer	老化时间，单位小时。
sync_replication	否	Boolean	是否同步复制。
sync_message_flush	否	Boolean	是否同步落盘。
new_partition_numbers	否	Integer	分区数。
new_partition_brokers	否	Array of integers	增加分区时指定broker列表
topic_other_configs	否	Array of **topic_other_configs** objects	topic配置
topic_desc	否	String	topic描述

表 5-137 topic_other_configs

参数	是否必选	参数类型	描述
name	否	String	配置名称
value	否	String	配置值

响应参数

无

请求示例

修改Topic参数，topic-1284340884的老化时间修改为72小时，分区数修改为6，新增分区分布在broker-1和broker-2上，不同步复制，不同步落盘，消息时间类型为LogAppendTime，最大批处理大小10485760。

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

{
  "topics" : [ {
    "id" : "test01",
    "retention_time" : 72,
    "sync_replication" : false,
    "sync_message_flush" : false,
    "new_partition_numbers" : 6,
    "new_partition_brokers" : [ 1, 2 ],
    "topic_other_configs" : [ {
```

```
    "name" : "message.timestamp.type",
    "value" : "LogAppendTime"
  }, {
    "name" : "max.message.bytes",
    "value" : 10485760
  } ],
  "topic_desc" : "This is a test topic"
 } ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改Topic参数，topic-1284340884的老化时间修改为72小时，分区数修改为6，新增分区分布在broker-1和broker-2上，不同步复制，不同步落盘，消息时间类型为LogAppendTime，最大批处理大小10485760。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class UpdateInstanceTopicSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceTopicRequest request = new UpdateInstanceTopicRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceTopicReq body = new UpdateInstanceTopicReq();
        List<UpdateInstanceTopicReqTopicOtherConfigs> listTopicsTopicOtherConfigs = new ArrayList<>();
        listTopicsTopicOtherConfigs.add(
            new UpdateInstanceTopicReqTopicOtherConfigs()
                .withName("message.timestamp.type")
```

```
                .withValue("LogAppendTime")
        );
        listTopicsTopicOtherConfigs.add(
            new UpdateInstanceTopicReqTopicOtherConfigs()
                .withName("max.message.bytes")
                .withValue("10485760")
        );
        List<Integer> listTopicsNewPartitionBrokers = new ArrayList<>();
        listTopicsNewPartitionBrokers.add(1);
        listTopicsNewPartitionBrokers.add(2);
        List<UpdateInstanceTopicReqTopics> listbodyTopics = new ArrayList<>();
        listbodyTopics.add(
            new UpdateInstanceTopicReqTopics()
                .withId("test01")
                .withRetentionTime(72)
                .withSyncReplication(false)
                .withSyncMessageFlush(false)
                .withNewPartitionNumbers(6)
                .withNewPartitionBrokers(listTopicsNewPartitionBrokers)
                .withTopicOtherConfigs(listTopicsTopicOtherConfigs)
                .withTopicDesc("This is a test topic")
        );
        body.withTopics(listbodyTopics);
        request.withBody(body);
        try {
            UpdateInstanceTopicResponse response = client.updateInstanceTopic(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Topic参数，topic-1284340884的老化时间修改为72小时，分区数修改为6，新增分区分布在broker-1和broker-2上，不同步复制，不同步落盘，消息时间类型为LogAppendTime，最大批处理大小10485760。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
```

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

    try:
        request = UpdateInstanceTopicRequest()
        request.instance_id = "{instance_id}"
        listTopicOtherConfigsTopics = [
            UpdateInstanceTopicReqTopicOtherConfigs(
                name="message.timestamp.type",
                value="LogAppendTime"
            ),
            UpdateInstanceTopicReqTopicOtherConfigs(
                name="max.message.bytes",
                value="10485760"
            )
        ]
        listNewPartitionBrokersTopics = [
            1,
            2
        ]
        listTopicsbody = [
            UpdateInstanceTopicReqTopics(
                id="test01",
                retention_time=72,
                sync_replication=False,
                sync_message_flush=False,
                new_partition_numbers=6,
                new_partition_brokers=listNewPartitionBrokersTopics,
                topic_other_configs=listTopicOtherConfigsTopics,
                topic_desc="This is a test topic"
            )
        ]
        request.body = UpdateInstanceTopicReq(
            topics=listTopicsbody
        )
        response = client.update_instance_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Topic参数，topic-1284340884的老化时间修改为72小时，分区数修改为6，新增分区分布在broker-1和broker-2上，不同步复制，不同步落盘，消息时间类型为LogAppendTime，最大批处理大小10485760。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceTopicRequest{}
    request.InstanceId = "{instance_id}"
    nameTopicOtherConfigs:= "message.timestamp.type"
    valueTopicOtherConfigs:= "LogAppendTime"
    nameTopicOtherConfigs1:= "max.message.bytes"
    valueTopicOtherConfigs1:= "10485760"
    var listTopicOtherConfigsTopics = []model.UpdateInstanceTopicReqTopicOtherConfigs{
        {
            Name: &nameTopicOtherConfigs,
            Value: &valueTopicOtherConfigs,
        },
        {
            Name: &nameTopicOtherConfigs1,
            Value: &valueTopicOtherConfigs1,
        },
    }
    var listNewPartitionBrokersTopics = []int32{
        int32(1),
        int32(2),
    }
    retentionTimeTopics:= int32(72)
    syncReplicationTopics:= false
    syncMessageFlushTopics:= false
    newPartitionNumbersTopics:= int32(6)
    topicDescTopics:= "This is a test topic"
    var listTopicsbody = []model.UpdateInstanceTopicReqTopics{
        {
            Id: "test01",
            RetentionTime: &retentionTimeTopics,
            SyncReplication: &syncReplicationTopics,
            SyncMessageFlush: &syncMessageFlushTopics,
            NewPartitionNumbers: &newPartitionNumbersTopics,
            NewPartitionBrokers: &listNewPartitionBrokersTopics,
            TopicOtherConfigs: &listTopicOtherConfigsTopics,
            TopicDesc: &topicDescTopics,
        },
    }
    request.Body = &model.UpdateInstanceTopicReq{
        Topics: &listTopicsbody,
    }
    response, err := client.UpdateInstanceTopic(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	修改成功。

错误码

请参见**错误码**。

5.5.5 Kafka 实例批量删除 Topic

功能介绍

该接口用于向Kafka实例批量删除Topic。批量删除多个Topic时，部分删除成功，部分失败，此时接口返回删除成功，并在返回中显示删除失败的Topic信息。

调用方法

请参见**如何调用API**。

URI

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

表 5-138 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 5-139 请求 Body 参数

参数	是否必选	参数类型	描述
topics	否	Array of strings	待删除的topic列表。 批量删除实例topic时，为必选参数。

响应参数

状态码：200

表 5-140 响应 Body 参数

参数	参数类型	描述
topics	Array of **topics** objects	Topic列表。

表 5-141 topics

参数	参数类型	描述
id	String	Topic名称。
success	Boolean	是否删除成功。

请求示例

批量删除Topic。

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

{
  "topics" : [ "topic01" ]
}
```

响应示例

状态码：200

删除成功。

```
{
  "topics" : [ {
    "id" : "topic01",
    "success" : true
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

批量删除Topic。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;
```

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

public class BatchDeleteInstanceTopicSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchDeleteInstanceTopicRequest request = new BatchDeleteInstanceTopicRequest();
        request.withInstanceId("{instance_id}");
        BatchDeleteInstanceTopicReq body = new BatchDeleteInstanceTopicReq();
        List<String> listbodyTopics = new ArrayList<>();
        listbodyTopics.add("topic01");
        body.withTopics(listbodyTopics);
        request.withBody(body);
        try {
            BatchDeleteInstanceTopicResponse response = client.batchDeleteInstanceTopic(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Topic。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"
```

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

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

try:
    request = BatchDeleteInstanceTopicRequest()
    request.instance_id = "{instance_id}"
    listTopicsbody = [
        "topic01"
    ]
    request.body = BatchDeleteInstanceTopicReq(
        topics=listTopicsbody
    )
    response = client.batch_delete_instance_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

批量删除Topic。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchDeleteInstanceTopicRequest{}
    request.InstanceId = "{instance_id}"
    var listTopicsbody = []string{
        "topic01",
    }
    request.Body = &model.BatchDeleteInstanceTopicReq{
        Topics: &listTopicsbody,
    }
```

```
    response, err := client.BatchDeleteInstanceTopic(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 Topic 的分区列表

功能介绍

查询Topic的分区列表

调用方法

请参见**如何调用API**。

URI

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

表 5-142 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

表 5-143 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	偏移量，表示查询该偏移量后面的记录
limit	否	Integer	查询返回记录的数量限制

请求参数

无

响应参数

状态码：200

表 5-144 响应 Body 参数

参数	参数类型	描述
total	Integer	总条数
partitions	Array of **partitions** objects	分区数组

表 5-145 partitions

参数	参数类型	描述
partition	Integer	分区ID
start_offset	Long	起始偏移量
last_offset	Long	最后偏移量
message_count	Long	分区消息数
last_update_time	Long	最近更新时间

请求示例

查询topic的分区列表

```
GET https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/topics/{topic}/partitions?start=1&limit=10
```

响应示例

状态码：200

查询topic的分区列表成功

```
{
  "total" : 3,
  "partitions" : [ {
    "partition" : 0,
    "start_offset" : 0,
    "last_offset" : 1216303,
    "message_count" : 1216303,
    "last_update_time" : 1688011291458
  }, {
    "partition" : 1,
    "start_offset" : 0,
    "last_offset" : 985447,
    "message_count" : 985447,
    "last_update_time" : 1688011291469
  }, {
    "partition" : 2,
    "start_offset" : 0,
    "last_offset" : 923340,
    "message_count" : 923340,
    "last_update_time" : 1688011291526
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListTopicPartitionsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListTopicPartitionsRequest request = new ListTopicPartitionsRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ListTopicPartitionsResponse response = client.listTopicPartitions(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListTopicPartitionsRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.list_topic_partitions(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListTopicPartitionsRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ListTopicPartitions(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topic的分区列表成功

错误码

请参见**错误码**。

5.5.7 查询 Topic 的当前生产者列表

功能介绍

查询Topic的当前生产者列表

调用方法

请参见**如何调用API**。

URI

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

表 5-146 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

表 5-147 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	偏移量，表示查询该偏移量后面的记录
limit	否	Integer	查询返回记录的数量限制

请求参数

无

响应参数

状态码：200

表 5-148 响应 Body 参数

参数	参数类型	描述
total	Integer	总条数
producers	Array of **producers** objects	生产者列表

表 5-149 producers

参数	参数类型	描述
producer_address	String	生产者地址
broker_address	String	broker地址
join_time	Long	加入时间

请求示例

查询Topic的当前生产者列表

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

响应示例

状态码：200

查询topic的当前生产者列表成功

```
{
  "total" : 3,
  "producers" : [ {
    "producer_address" : "192.0.0.149:40443",
    "broker_address" : "192.0.0.146:9092",
    "join_time" : 1687204743328
  }, {
    "producer_address" : "192.0.0.149:13807",
    "broker_address" : "192.0.0.80:9092",
    "join_time" : 1687204745939
  }, {
    "producer_address" : "192.0.0.149:31876",
    "broker_address" : "192.0.0.71:9092",
    "join_time" : 1687204744934
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListTopicProducersSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
```

```
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListTopicProducersRequest request = new ListTopicProducersRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ListTopicProducersResponse response = client.listTopicProducers(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListTopicProducersRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.list_topic_producers(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"
```

```
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListTopicProducersRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ListTopicProducers(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topic的当前生产者列表成功

错误码

请参见**错误码**。

5.5.8 查询 Kafka 实例 Topic 详细信息

功能介绍

查询Kafka实例Topic详细信息。(单个实例调用不要超过1s一次)

调用方法

请参见**如何调用API**。

URI

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

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。
topic	是	String	Topic名称。

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
topic	String	topic名称。
partitions	Array of **partitions** objects	分区列表。
group_subscribed	Array of strings	订阅该topic的消费组名称列表。

表 5-152 partitions

参数	参数类型	描述
partition	Integer	分区ID。
leader	Integer	leader副本所在节点的id。
leo	Integer	分区leader副本的LEO（Log End Offset）。
hw	Integer	分区高水位（HW，High Watermark）。

参数	参数类型	描述
lso	Integer	分区leader副本的LSO（Log Start Offset）。
last_update_timestamp	Long	分区上次写入消息的时间。 格式为Unix时间戳。 单位：毫秒。
replicas	Array of **replicas** objects	副本列表。

表 5-153 replicas

参数	参数类型	描述
broker	Integer	副本所在的节点ID。
leader	Boolean	该副本是否为leader。
in_sync	Boolean	该副本是否在ISR副本中。
size	Integer	该副本当前日志大小。单位：Byte。
lag	Long	该副本当前落后hw的消息数。

请求示例

查询指定Topic的详细信息。

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

响应示例

状态码：200

查询成功。

```
{
  "topic" : "test",
  "partitions" : [ {
    "partition" : 0,
    "leader" : 2,
    "replicas" : [ {
      "broker" : 2,
      "leader" : true,
      "in_sync" : true,
      "size" : 123971146,
      "lag" : 0
    }, {
      "broker" : 1,
      "leader" : false,
      "in_sync" : true,
      "size" : 123971146,
      "lag" : 0
    }, {
      "broker" : 0,
```

```
    "leader" : false,
    "in_sync" : true,
    "size" : 123971146,
    "lag" : 0
   } ],
   "lso" : 0,
   "leo" : 13598,
   "hw" : 13598,
   "last_update_timestamp" : 1571477180985
  }, {
   "partition" : 2,
   "leader" : 1,
   "replicas" : [ {
    "broker" : 1,
    "leader" : true,
    "in_sync" : true,
    "size" : 123889531,
    "lag" : 0
   }, {
    "broker" : 0,
    "leader" : false,
    "in_sync" : true,
    "size" : 123889531,
    "lag" : 0
   }, {
    "broker" : 2,
    "leader" : false,
    "in_sync" : true,
    "size" : 123889531,
    "lag" : 0
   } ],
   "lso" : 0,
   "leo" : 13601,
   "hw" : 13601,
   "last_update_timestamp" : 1571477077146
  }, {
   "partition" : 1,
   "leader" : 0,
   "replicas" : [ {
    "broker" : 0,
    "leader" : true,
    "in_sync" : true,
    "size" : 127245604,
    "lag" : 0
   }, {
    "broker" : 2,
    "leader" : false,
    "in_sync" : true,
    "size" : 127245604,
    "lag" : 0
   }, {
    "broker" : 1,
    "leader" : false,
    "in_sync" : true,
    "size" : 127245604,
    "lag" : 0
   } ],
   "lso" : 0,
   "leo" : 13599,
   "hw" : 13599,
   "last_update_timestamp" : 1571477172959
  } ],
  "group_subscribed" : [ "test-consumer-group" ]
}
```


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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowInstanceTopicDetailSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowInstanceTopicDetailRequest request = new ShowInstanceTopicDetailRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ShowInstanceTopicDetailResponse response = client.showInstanceTopicDetail(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowInstanceTopicDetailRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.show_instance_topic_detail(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowInstanceTopicDetailRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ShowInstanceTopicDetail(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**。

5.6 消费组管理

5.6.1 查询消费组信息

功能介绍

查询消费组信息。

调用方法

请参见**如何调用API**。

URI

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

表 5-154 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 5-155 响应 Body 参数

参数	参数类型	描述
group	**group** object	消费组信息。

表 5-156 group

参数	参数类型	描述
group_id	String	消费组名称。
state	String	消费组状态。包含以下状态： • Dead：消费组内没有任何成员，且没有任何元数据。 • Empty：消费组内没有任何成员，存在元数据。 • PreparingRebalance：准备开启rebalance。 • CompletingRebalance：所有成员加入group。 • Stable：消费组内成员可正常消费。
coordinator_id	Integer	协调器编号。
members	Array of **members** objects	消费者列表。
group_message_offsets	Array of **group_message_offsets** objects	消费进度。
assignment_strategy	String	分区分配策略。

表 5-157 members

参数	参数类型	描述
host	String	消费组consumer地址。
assignment	Array of **assignment** objects	consumer分配到的分区信息。
member_id	String	消费组consumer的ID。

参数	参数类型	描述
client_id	String	客户端ID。

表 5-158 assignment

参数	参数类型	描述
topic	String	topic名称。
partitions	Array of integers	分区列表。

表 5-159 group_message_offsets

参数	参数类型	描述
partition	Integer	分区编号。
lag	Long	剩余可消费消息数，即消息堆积数。
topic	String	topic名称。
message_current_offset	Long	当前消费进度。
message_log_end_offset	Long	最大消息位置（LEO）。

请求示例

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

响应示例

状态码：200

查询实例集群的消费组信息成功。

```
{
  "group" : {
    "members" : [ {
      "host" : "/172.31.1.102",
      "assignment" : [ {
        "topic" : "test",
        "partitions" : [ 0, 1, 2 ]
      } ],
      "member_id" : "consumer-1-6b8ee551-d499-47d4-9beb-ba1527496785",
      "client_id" : "consumer-1"
    } ],
    "state" : "STABLE",
    "group_id" : "test-consumer-group",
    "coordinator_id" : 2,
    "group_message_offsets" : [ {
      "partition" : 0,
      "lag" : 31396,
```

```
      "topic" : "test",
      "message_current_offset" : 935,
      "message_log_end_offset" : 32331
    }, {
      "partition" : 0,
      "lag" : 0,
      "topic" : "aaaa",
      "message_current_offset" : 0,
      "message_log_end_offset" : 0
    }, {
      "partition" : 1,
      "lag" : 31279,
      "topic" : "test",
      "message_current_offset" : 1058,
      "message_log_end_offset" : 32337
    }, {
      "partition" : 1,
      "lag" : 0,
      "topic" : "aaaa",
      "message_current_offset" : 0,
      "message_log_end_offset" : 0
    }, {
      "partition" : 2,
      "lag" : 31603,
      "topic" : "test",
      "message_current_offset" : 739,
      "message_log_end_offset" : 32342
    } ],
    "assignment_strategy" : "range"
  }
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowGroupsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
```

```
            .withCredential(auth)
            .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowGroupsRequest request = new ShowGroupsRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        try {
            ShowGroupsResponse response = client.showGroups(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowGroupsRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        response = client.show_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"
```

```
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowGroupsRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    response, err := client.ShowGroups(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	查询实例集群的消费组信息成功。

错误码

请参见**错误码**。

5.6.2 查询所有消费组

功能介绍

查询所有消费组。

调用方法

请参见**如何调用API**。

URI

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

表 5-160 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 5-161 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	当次查询返回的最大消费组ID个数，默认值为10，取值范围为1~50。
group	否	String	消费组名过滤查询，过滤方式为字段包含过滤。

请求参数

无

响应参数

状态码：200

表 5-162 响应 Body 参数

参数	参数类型	描述
groups	Array of **GroupInfoSimple** objects	所有的消费组。
total	Integer	所有的消费组总数。

表 5-163 GroupInfoSimple

参数	参数类型	描述
createdAt	Long	创建时间。
group_id	String	消费组ID。
state	String	消费组状态。包含以下状态： ● Dead：消费组内没有任何成员，且没有任何元数据。 ● Empty：消费组内没有任何成员，存在元数据。 ● PreparingRebalance：准备开启rebalance。 ● CompletingRebalance：所有成员加入group。 ● Stable：消费组内成员可正常消费。
coordinator_id	Integer	协调器编号。
group_desc	String	消费组描述。
lag	Long	堆积数。

请求示例

查询消费组列表。

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/groups?
offset={offset}&limit={limit}&group={group}
```

响应示例

状态码：200

查询实例集群的所有消费组成功。

```
{
  "groups" : [ {
    "createdAt" : 1691401194847,
    "group_id" : "consumer-1",
    "state" : "EMPTY",
    "coordinator_id" : 1,
    "lag" : 0,
    "group_desc" : null
  }, {
    "createdAt" : 1691401194960,
    "group_id" : "consumer-2",
    "state" : "STABLE",
    "coordinator_id" : 2,
    "lag" : 0,
    "group_desc" : null
  }, {
    "createdAt" : 1691401207309,
    "group_id" : "consumer-3",
    "state" : "STABLE",
    "coordinator_id" : 3,
```

```
    "lag" : 0,
    "group_desc" : null
  } ],
  "total" : 3
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.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
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            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	查询实例集群的所有消费组成功。

错误码

请参见**错误码**。

5.6.3 Kafka 实例批量删除消费组

功能介绍

该接口用于向Kafka实例批量删除消费组。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/groups/batch-delete

表 5-164 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 5-165 请求 Body 参数

参数	是否必选	参数类型	描述
group_ids	是	Array of strings	所有需要删除的消费组ID。

响应参数

状态码：200

表 5-166 响应 Body 参数

参数	参数类型	描述
failed_groups	Array of **failed_groups** objects	删除失败的消费组列表。
total	Integer	删除失败的个数

表 5-167 failed_groups

参数	参数类型	描述
group_id	String	删除失败的消费组ID。
error_message	String	删除失败的原因。

请求示例

批量删除消费组。

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

{
  "group_ids" : [ "get-sync-group0", "get-sync-group1" ]
}
```

响应示例

状态码：200

删除消费组成功。

```
{
  "failed_groups" : [ {
    "group_id" : "test-1",
    "error_message" : "UNKNOW"
  }, {
```

```
    "group_id" : "test-2",
    "error_message" : "UNKNOW"
  } ],
  "total" : 2
}
```

SDK 代码示例

SDK代码示例如下。

Java

批量删除消费组。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class BatchDeleteGroupSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchDeleteGroupRequest request = new BatchDeleteGroupRequest();
        request.withInstanceId("{instance_id}");
        BatchDeleteGroupReq body = new BatchDeleteGroupReq();
        List<String> listbodyGroupIds = new ArrayList<>();
        listbodyGroupIds.add("get-sync-group0");
        listbodyGroupIds.add("get-sync-group1");
        body.withGroupIds(listbodyGroupIds);
        request.withBody(body);
        try {
            BatchDeleteGroupResponse response = client.batchDeleteGroup(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

批量删除消费组。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchDeleteGroupRequest()
        request.instance_id = "{instance_id}"
        listGroupIdsbody = [
            "get-sync-group0",
            "get-sync-group1"
        ]
        request.body = BatchDeleteGroupReq(
            group_ids=listGroupIdsbody
        )
        response = client.batch_delete_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

批量删除消费组。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchDeleteGroupRequest{}
    request.InstanceId = "{instance_id}"
    var listGroupIdsbody = []string{
        "get-sync-group0",
        "get-sync-group1",
    }
    request.Body = &model.BatchDeleteGroupReq{
        GroupIds: listGroupIdsbody,
    }
    response, err := client.BatchDeleteGroup(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	删除消费组成功。

错误码

请参见**错误码**。

5.6.4 创建消费组

功能介绍

实例创建消费组

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/kafka/instances/{instance_id}/group

表 5-168 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 5-169 请求 Body 参数

参数	是否必选	参数类型	描述
group_name	是	String	消费组名称
group_desc	否	String	消费组描述

响应参数

状态码：200

表 5-170 响应 Body 参数

参数	参数类型	描述
-	String	创建结果

状态码：400

表 5-171 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

请求示例

创建一个消费组，消费组名为test。

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

{
  "group_name" : "test"
}
```

响应示例

状态码：200

创建成功。

```
success
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个消费组，消费组名为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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateKafkaConsumerGroupSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateKafkaConsumerGroupRequest request = new CreateKafkaConsumerGroupRequest();
        request.withInstanceId("{instance_id}");
        CreateGroupReq body = new CreateGroupReq();
        body.withGroupName("test");
        request.withBody(body);
        try {
            CreateKafkaConsumerGroupResponse response = client.createKafkaConsumerGroup(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

创建一个消费组，消费组名为test。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateKafkaConsumerGroupRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateGroupReq(
            group_name="test"
        )
        response = client.create_kafka_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

创建一个消费组，消费组名为test。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateKafkaConsumerGroupRequest{}
    request.InstanceId = "{instance_id}"
    request.Body = &model.CreateGroupReq{
        GroupName: "test",
    }
    response, err := client.CreateKafkaConsumerGroup(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。
400	创建失败。

错误码

请参见**错误码**。

5.6.5 重置消费组消费进度到指定位置

功能介绍

Kafka实例不支持在线重置消费进度。在执行重置消费进度之前，必须停止被重置消费组客户端。停止待重置消费组客户端，然后等待一段时间（即

ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG配置的时间，默认为1000毫秒）后，服务端才认为此消费组客户端已下线。

调用方法

请参见**如何调用API**。

URI

PUT /v2/kafka/{project_id}/instances/{instance_id}/groups/{group}/reset-message-offset

表 5-172 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID。
instance_id	是	String	实例ID。
group	是	String	消费组名称。

请求参数

表 5-173 请求 Body 参数

参数	是否必选	参数类型	描述
topic	否	String	topic名称。
partition	是	Integer	分区编号，默认值为-1，若传入值为-1，则重置所有分区。
message_offset	否	Long	重置消费进度到指定偏移量。 • 如果传入offset小于当前最小的offset，则重置到最小的offset。 • 如果大于最大的offset，则重置到最大的offset。 **message_offset、timestamp二者必选其一。**

参数	是否必选	参数类型	描述
timestamp	否	Long	重置消费进度到指定时间，格式为unix时间戳，单位为毫秒。 ● 如果传入timestamp早于当前最早的timestamp，则重置到最早的timestamp。 ● 如果晚于最晚的timestamp，则重置到最晚的timestamp。 **message_offset、timestamp二者必选其一。**

响应参数

无

请求示例

- 重置的消费进度到指定偏移量。

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

{
  "topic" : "test",
  "partition" : 0,
  "message_offset" : 10
}
```

- 重置的消费进度到指定时间。

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

{
  "topic" : "test",
  "partition" : 0,
  "timestamp" : 1571812144000
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

- 重置的消费进度到指定偏移量。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetMessageOffsetWithEngineSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetMessageOffsetWithEngineRequest request = new ResetMessageOffsetWithEngineRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        ResetMessageOffsetReq body = new ResetMessageOffsetReq();
        body.withMessageOffset(10L);
        body.withPartition(0);
        body.withTopic("test");
        request.withBody(body);
        try {
            ResetMessageOffsetWithEngineResponse response = client.resetMessageOffsetWithEngine(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());
        }
    }
}
```

- 重置的消费进度到指定时间。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetMessageOffsetWithEngineSolution {
```

```
    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetMessageOffsetWithEngineRequest request = new ResetMessageOffsetWithEngineRequest();
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        ResetMessageOffsetReq body = new ResetMessageOffsetReq();
        body.withTimestamp(1571812144000L);
        body.withPartition(0);
        body.withTopic("test");
        request.withBody(body);
        try {
            ResetMessageOffsetWithEngineResponse response = client.resetMessageOffsetWithEngine(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

- 重置的消费进度到指定偏移量。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

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



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

    try:
        request = ResetMessageOffsetWithEngineRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        request.body = ResetMessageOffsetReq(
            message_offset=10,
            partition=0,
            topic="test"
        )
        response = client.reset_message_offset_with_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)
```

- 重置的消费进度到指定时间。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResetMessageOffsetWithEngineRequest()
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        request.body = ResetMessageOffsetReq(
            timestamp=1571812144000,
            partition=0,
            topic="test"
        )
        response = client.reset_message_offset_with_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

- 重置的消费进度到指定偏移量。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetMessageOffsetWithEngineRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    messageOffsetResetMessageOffsetReq:= int64(10)
    topicResetMessageOffsetReq:= "test"
    request.Body = &model.ResetMessageOffsetReq{
        MessageOffset: &messageOffsetResetMessageOffsetReq,
        Partition: int32(0),
        Topic: &topicResetMessageOffsetReq,
    }
    response, err := client.ResetMessageOffsetWithEngine(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 重置的消费进度到指定时间。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetMessageOffsetWithEngineRequest{}
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    timestampResetMessageOffsetReq:= int64(1571812144000)
    topicResetMessageOffsetReq:= "test"
    request.Body = &model.ResetMessageOffsetReq{
        Timestamp: &timestampResetMessageOffsetReq,
        Partition: int32(0),
        Topic: &topicResetMessageOffsetReq,
    }
    response, err := client.ResetMessageOffsetWithEngine(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	重置消费组消息进度到指定位置操作成功。

错误码

请参见**错误码**。

5.6.6 查询消费组消息位点

功能介绍

查询消费组消息位点。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/{project_id}/instances/{instance_id}/groups/{group}/message-offset

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。
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 5-175 Query 参数

参数	是否必选	参数类型	描述
topic	否	String	topic名称。
partition	否	String	分区名称。
offset	否	String	偏移值。
limit	否	String	最大值。

请求参数

无

响应参数

状态码：200

表 5-176 响应 Body 参数

参数	参数类型	描述
group_message_offsets	Array of **GroupMessageOffsetsDetailEntity** objects	消费组消息位点详情
total	Integer	总数

表 5-177 GroupMessageOffsetsDetailEntity

参数	参数类型	描述
partition	String	分区
message_current_offset	String	消息当前位点
message_log_start_offset	Integer	消息开始位点
message_log_end_offset	Integer	消息结束位点
consumer_id	String	消费者Id
host	String	host名称
client_id	String	客户端Id

请求示例

查询消费组消息位点

```
GET https://{endpoint}/v2/{engine}/{project_id}/instances/{instance_id}/groups/{group}/message-offset
```

响应示例

状态码：200

查询成功

```
{
  "group_message_offsets" : [ ],
  "total" : 0
}
```

状态码

状态码	描述
200	查询成功

错误码

请参见**错误码**。

5.6.7 修改所有消费组

功能介绍

修改所有消费组。

调用方法

请参见**如何调用API**。

URI

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

表 5-178 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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 5-179 请求 Body 参数

参数	是否必选	参数类型	描述
group_name	否	String	消费组名称
group_desc	否	String	消费组描述

响应参数

状态码：200

表 5-180 响应 Body 参数

参数	参数类型	描述
job_id	String	后台任务ID

请求示例

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

响应示例

状态码：200

操作成功。

```
{
  "job_id" : "8a2c259182ab0e9d0182ab1882560010"
}
```

状态码

状态码	描述
200	操作成功。

错误码

请参见**错误码**。

5.6.8 查询指定消费组

功能介绍

查询指定消费组。

调用方法

请参见**如何调用API**。

URI

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

表 5-181 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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	消费组名过滤查询。

请求参数

无

响应参数

状态码：200

表 5-182 响应 Body 参数

参数	参数类型	描述
group	**group** object	消费组信息。

表 5-183 group

参数	参数类型	描述
group_id	String	消费组名称。
state	String	消费组状态。包含以下状态： • Dead：消费组内没有任何成员，且没有任何元数据。 • Empty：消费组内没有任何成员，存在元数据。 • PreparingRebalance：准备开启rebalance。 • CompletingRebalance：所有成员加入group。 • Stable：消费组内成员可正常消费。
coordinator_id	Integer	协调器编号。
members	Array of **members** objects	消费者列表。
group_message_offsets	Array of **group_message_offsets** objects	消费进度。
assignment_strategy	String	分区分配策略。

表 5-184 members

参数	参数类型	描述
host	String	消费组consumer地址。
member_id	String	消费组consumer的ID。
client_id	String	客户端ID。

表 5-185 group_message_offsets

参数	参数类型	描述
partition	Integer	分区编号。
lag	Long	剩余可消费消息数，即消息堆积数。
topic	String	topic名称。
message_current_offset	Long	当前消费进度。
message_log_end_offset	Long	最大消息位置（LEO）。

请求示例

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

响应示例

状态码：200

查询指定消费组信息成功。

```
{
  "group" : null,
  "members" : [ {
    "host" : "/172.31.1.102",
    "member_id" : "consumer-1-6b8ee551-d499-47d4-9beb-ba1527496785",
    "client_id" : "consumer-1"
  } ],
  "state" : "STABLE",
  "group_id" : "test-consumer-group",
  "coordinator_id" : 2,
  "group_message_offsets" : [ {
    "partition" : 0,
    "lag" : 31396,
    "topic" : "test",
    "message_current_offset" : 935,
    "message_log_end_offset" : 32331
  }, {
    "partition" : 0,
    "lag" : 0,
    "topic" : "aaaa",
    "message_current_offset" : 0,
    "message_log_end_offset" : 0
  }, {
    "partition" : 1,
    "lag" : 31279,
    "topic" : "test",
    "message_current_offset" : 1058,
    "message_log_end_offset" : 32337
  }, {
    "partition" : 1,
    "lag" : 0,
    "topic" : "aaaa",
    "message_current_offset" : 0,
    "message_log_end_offset" : 0
  }, {
    "partition" : 2,
    "lag" : 31603,
```

```
    "topic" : "test",
    "message_current_offset" : 739,
    "message_log_end_offset" : 32342
  } ],
  "assignment_strategy" : "range"
}
```

状态码

状态码	描述
200	查询指定消费组信息成功。

错误码

请参见**错误码**。

5.6.9 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/{engine}/{project_id}/instances/{instance_id}/groups/{group}

表 5-186 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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ID。

请求参数

无

响应参数

无

请求示例

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

响应示例

无

状态码

状态码	描述
200	删除成功。

错误码

请参见**错误码**。

5.6.10 修改指定消费组

功能介绍

修改指定消费组。

调用方法

请参见**如何调用API**。

URI

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

表 5-187 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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ID。

请求参数

表 5-188 请求 Body 参数

参数	是否必选	参数类型	描述
group_name	否	String	消费组名称
group_desc	否	String	消费组描述

响应参数

无

请求示例

```
PUT https://{endpoint}/v2/{engine}/{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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class UpdateInstanceConsumerGroupSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
```

```
                .build();
        UpdateInstanceConsumerGroupRequest request = new UpdateInstanceConsumerGroupRequest();
        request.withEngine("{engine}");
        request.withInstanceId("{instance_id}");
        request.withGroup("{group}");
        CreateGroupReq body = new CreateGroupReq();
        request.withBody(body);
        try {
            UpdateInstanceConsumerGroupResponse response = client.updateInstanceConsumerGroup(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceConsumerGroupRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.group = "{group}"
        request.body = CreateGroupReq(
        )
        response = client.update_instance_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceConsumerGroupRequest{}
    request.Engine = "{engine}"
    request.InstanceId = "{instance_id}"
    request.Group = "{group}"
    request.Body = &model.CreateGroupReq{
    }
    response, err := client.UpdateInstanceConsumerGroup(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	操作成功。

错误码

请参见**错误码**。

5.6.11 查询指定消费组的 Topic

功能介绍

查询指定消费组的Topic。

调用方法

请参见**如何调用API**。

URI

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

表 5-189 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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ID。

表 5-190 Query 参数

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。
limit	否	Integer	当次查询返回的最大topic个数，默认值为10，取值范围为1~50。
sort_key	否	String	排序规则： • topic：按topic名称排序。 • partition：按分区数排序。 • messages：按消息数量排序，默认方式。
sort_dir	否	String	排序方式。 • asc：升序。 • desc：降序，默认方式。
topic	否	String	topic名称。

请求参数

无

响应参数

状态码：200

表 5-191 响应 Body 参数

参数	参数类型	描述
topics	Array of **GroupTopicEntity** objects	消费组TOPIC
total	Integer	统计数量

表 5-192 GroupTopicEntity

参数	参数类型	描述
topic	String	TOPIC名称
partitions	Integer	分区
lag	Integer	消息堆积数量

请求示例

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

响应示例

状态码：200

查询成功。

```
{
  "topics" : [ {
    "topic" : "topic-1",
    "partitions" : 1,
    "lag" : 0
  }, {
    "topic" : "topic-2",
    "partitions" : 2,
    "lag" : 10
  } ],
  "total" : 2
}
```

状态码

状态码	描述
200	查询成功。

错误码

请参见**错误码**。

5.6.12 查询指定消费组的消费成员

功能介绍

查询指定消费组的消费成员。

调用方法

请参见**如何调用API**。

URI

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

表 5-193 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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ID。

表 5-194 Query 参数

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。
limit	否	Integer	当次查询返回的最大消费组成员个数，默认值为10，取值范围为1~50。
host	否	String	消费者地址。
member_id	否	String	消费者ID。

请求参数

无

响应参数

状态码：200

表 5-195 响应 Body 参数

参数	参数类型	描述
members	Array of **GroupMemberEntity** objects	成员详情
total	Integer	总数

表 5-196 GroupMemberEntity

参数	参数类型	描述
member_id	String	成员Id
client_id	String	客户端Id

请求示例

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

响应示例

状态码：200

查询成功。

```
{
  "members" : [ {
    "member_id" : "consumer-1-6b8ee551-d499-47d4-9beb-ba1527496785",
    "client_id" : "consumer-1"
  }, {
    "member_id" : "consumer-2-6b8ee551-d499-47d4-9beb-ba1527491125",
    "client_id" : "consumer-2"
  } ],
  "total" : 2
}
```

状态码

状态码	描述
200	查询成功。

错误码

请参见**错误码**。

5.7 用户管理

5.7.1 查询用户列表

功能介绍

查询用户列表。

Kafka实例开启SASL功能时，才支持多用户管理的功能。

调用方法

请参见**如何调用API**。

URI

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

表 5-197 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 5-198 响应 Body 参数

参数	参数类型	描述
users	Array of **ShowInstanceUsersEntity** objects	用户列表。

表 5-199 ShowInstanceUsersEntity

参数	参数类型	描述
user_name	String	用户名称。 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。
user_desc	String	用户描述。
role	String	用户角色。
default_app	Boolean	是否为默认应用。
created_time	Long	创建时间。

请求示例

查询用户列表。

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

响应示例

状态码：200

查询成功。

```
{
  "users" : [ {
    "user_name" : "xxxa",
    "role" : "guest",
    "default_app" : false,
    "created_time" : 1615431764734
  }, {
    "user_name" : "test",
    "role" : "guest",
    "default_app" : false,
    "created_time" : 1615364062463
  }, {
    "user_name" : "ROOT",
    "role" : "guest",
    "default_app" : false,
    "created_time" : 1617194246328
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowInstanceUsersSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowInstanceUsersRequest request = new ShowInstanceUsersRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowInstanceUsersResponse response = client.showInstanceUsers(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowInstanceUsersRequest()
        request.instance_id = "{instance_id}"
        response = client.show_instance_users(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowInstanceUsersRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowInstanceUsers(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**。

5.7.2 创建用户

功能介绍

创建Kafka实例的用户，用户可连接开启SASL的Kafka实例。 2023年7月15日前创建的Kafka实例，一个实例最多创建20个用户。2023年7月15日及以后创建的Kafka实例，一个实例最多创建500个用户。

调用方法

请参见**如何调用API**。

URI

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

表 5-200 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 5-201 请求 Body 参数

参数	是否必选	参数类型	描述
user_name	否	String	用户名称。 创建用户时，为必选参数。
user_desc	否	String	用户描述。
user_passwd	否	String	用户密码。 创建用户时，为必选参数。 密码不能和用户名相同。 复杂度要求： • 输入长度为8到32位的字符串。 • 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（`~!@#$%^&*()-_=+|[{}]:'",<.>/?）和空格，并且不能以-开头

响应参数

状态码：400

表 5-202 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

状态码：403

表 5-203 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。

参数	参数类型	描述
error_msg	String	错误描述。

请求示例

创建一个用户，用户名为test，密码为Cxxx3。

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

{
  "user_name" : "test",
  "user_passwd" : "Cxxx3"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

创建一个用户，用户名为test，密码为Cxxx3。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateInstanceUserSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateInstanceUserRequest request = new CreateInstanceUserRequest();
```

```
        request.withInstanceId("{instance_id}");
        CreateInstanceUserReq body = new CreateInstanceUserReq();
        body.withUserPasswd("Cxxx3");
        body.withUserName("test");
        request.withBody(body);
        try {
            CreateInstanceUserResponse response = client.createInstanceUser(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

创建一个用户，用户名为test，密码为Cxxx3。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateInstanceUserRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateInstanceUserReq(
            user_passwd="Cxxx3",
            user_name="test"
        )
        response = client.create_instance_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

创建一个用户，用户名为test，密码为Cxxx3。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateInstanceUserRequest{}
    request.InstanceId = "{instance_id}"
    userPasswdCreateInstanceUserReq:= "Cxxx3"
    userNameCreateInstanceUserReq:= "test"
    request.Body = &model.CreateInstanceUserReq{
        UserPasswd: &userPasswdCreateInstanceUserReq,
        UserName: &userNameCreateInstanceUserReq,
    }
    response, err := client.CreateInstanceUser(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	创建成功。
400	参数无效。

状态码	描述
403	鉴权失败。

错误码

请参见**错误码**。

5.7.3 批量删除用户

功能介绍

批量删除Kafka实例的用户。

调用方法

请参见**如何调用API**。

URI

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

表 5-204 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 5-205 请求 Body 参数

参数	是否必选	参数类型	描述
action	否	String	删除类型。当前只支持delete。 当删除用户时，为必选参数。
users	否	Array of strings	用户列表。 当删除用户时，为必选参数。

响应参数

无

请求示例

批量删除用户。

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

{
  "action" : "delete",
  "users" : [ "testuser" ]
}
```

响应示例

无

状态码

状态码	描述
204	删除成功。

错误码

请参见**错误码**。

5.7.4 重置用户密码

功能介绍

重置用户密码

调用方法

请参见**如何调用API**。

URI

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

表 5-206 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	用户名称。

请求参数

表 5-207 请求 Body 参数

参数	是否必选	参数类型	描述
new_password	否	String	用户新密码。 重置用户密码时，为必选参数； 不能与名称或倒序的名称相同。 复杂度要求： ● 输入长度为8到32位的字符串。 ● 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（ `~!@#$%^&*()-_=+|[{}]:'",<.>/? ）和空格，并且不能以-开头

响应参数

无

请求示例

重置用户密码。

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

{
  "new_password" : "Cxxx3"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

重置用户密码。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetUserPasswrodSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetUserPasswrodRequest request = new ResetUserPasswrodRequest();
        request.withInstanceId("{instance_id}");
        request.withUserName("{user_name}");
        ResetUserPasswrodReq body = new ResetUserPasswrodReq();
        body.withNewPassword("Cxxx3");
        request.withBody(body);
        try {
            ResetUserPasswrodResponse response = client.resetUserPasswrod(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

重置用户密码。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ResetUserPasswrodRequest()
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        request.body = ResetUserPasswrodReq(
            new_password="Cxxx3"
        )
        response = client.reset_user_passwrod(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

重置用户密码。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetUserPasswrodRequest{}
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    newPasswordResetUserPasswrodReq:= "Cxxx3"
    request.Body = &model.ResetUserPasswrodReq{
```

```
        NewPassword: &newPasswordResetUserPasswrodReq,
    }
    response, err := client.ResetUserPasswrod(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	重置成功。

错误码

请参见**错误码**。

5.7.5 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/{engine}/{project_id}/instances/{instance_id}/users/{user_name}

表 5-208 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。
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	用户名称。

请求参数

表 5-209 请求 Body 参数

参数	是否必选	参数类型	描述
new_password	否	String	用户新密码。 不能与名称或倒序的名称相同。 复杂度要求： ● 输入长度为8到32位的字符串。 ● 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（`~!@#$%^&*()-_=+|[{}]:'",<.>/?）和空格，并且不能以-开头
user_name	否	String	用户名。
user_desc	否	String	用户描述。

响应参数

无

请求示例

修改用户参数。

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

{
  "new_password" : "Cxxx3",
  "user_name" : "用户名",
  "user_desc" : "用户描述"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改用户参数。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class UpdateInstanceUserSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateInstanceUserRequest request = new UpdateInstanceUserRequest();
        request.withEngine("{engine}");
        request.withInstanceId("{instance_id}");
        request.withUserName("{user_name}");
        UpdateUserReq body = new UpdateUserReq();
        body.withUserDesc("用户描述");
        body.withUserName("用户名");
        body.withNewPassword("Cxxx3");
        request.withBody(body);
        try {
            UpdateInstanceUserResponse response = client.updateInstanceUser(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。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateInstanceUserRequest()
        request.engine = "{engine}"
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        request.body = UpdateUserReq(
            user_desc="用户描述",
            user_name="用户名",
            new_password="Cxxx3"
        )
        response = client.update_instance_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。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
```

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

    request := &model.UpdateInstanceUserRequest{}
    request.Engine = "{engine}"
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    userDescUpdateUserReq:= "用户描述"
    userNameUpdateUserReq:= "用户名"
    newPasswordUpdateUserReq:= "Cxxx3"
    request.Body = &model.UpdateUserReq{
        UserDesc: &userDescUpdateUserReq,
        UserName: &userNameUpdateUserReq,
        NewPassword: &newPasswordUpdateUserReq,
    }
    response, err := client.UpdateInstanceUser(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	修改成功。

错误码

请参见**错误码**。

5.7.6 查询用户权限

功能介绍

查询用户权限。

Kafka实例开启SASL功能时，才支持多用户管理的功能。

调用方法

请参见**如何调用API**。

URI

GET /v1/{project_id}/instances/{instance_id}/topics/{topic_name}/accesspolicy

表 5-210 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_name	是	String	Topic名称。

请求参数

无

响应参数

状态码：200

表 5-211 响应 Body 参数

参数	参数类型	描述
name	String	topic名称。
topic_type	Integer	topic类型。
policies	Array of **PolicyEntity** objects	权限列表。

表 5-212 PolicyEntity

参数	参数类型	描述
owner	Boolean	是否为创建topic时所选择的用户。
user_name	String	用户名。
access_policy	String	权限类型。 • all：拥有发布、订阅权限; • pub：拥有发布权限; • sub：拥有订阅权限。

请求示例

查询Topic的用户权限。

```
GET https://{endpoint}/v1/{project_id}/instances/{instance_id}/topics/{topic_name}/accesspolicy
```

响应示例

状态码：200

查询成功。

```
{
  "name" : "topic-test",
  "policies" : [ {
    "owner" : false,
    "user_name" : "xxxa",
    "access_policy" : "pub"
  }, {
    "owner" : false,
    "user_name" : "root",
    "access_policy" : "all"
  } ],
  "topic_type" : 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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowTopicAccessPolicySolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowTopicAccessPolicyRequest request = new ShowTopicAccessPolicyRequest();
        request.withInstanceId("{instance_id}");
        request.withTopicName("{topic_name}");
        try {
            ShowTopicAccessPolicyResponse response = client.showTopicAccessPolicy(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowTopicAccessPolicyRequest()
        request.instance_id = "{instance_id}"
        request.topic_name = "{topic_name}"
        response = client.show_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowTopicAccessPolicyRequest{}
    request.InstanceId = "{instance_id}"
    request.TopicName = "{topic_name}"
    response, err := client.ShowTopicAccessPolicy(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**。

5.7.7 设置用户权限

功能介绍

设置用户权限。

Kafka实例开启SASL功能时，才支持多用户管理的功能。

调用方法

请参见**如何调用API**。

URI

POST /v1/{project_id}/instances/{instance_id}/topics/accesspolicy

表 5-213 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 5-214 请求 Body 参数

参数	是否必选	参数类型	描述
topics	是	Array of **AccessPolicyTopicEntity** objects	topic列表。

表 5-215 AccessPolicyTopicEntity

参数	是否必选	参数类型	描述
name	是	String	topic名称。
policies	是	Array of **AccessPolicyEntity** objects	权限列表。

表 5-216 AccessPolicyEntity

参数	是否必选	参数类型	描述
user_name	否	String	用户名称。 设置用户权限时，为必选参数。
access_policy	否	String	权限类型。 • all：拥有发布、订阅权限; • pub：拥有发布权限; • sub：拥有订阅权限。 设置用户权限时，为必选参数。

响应参数

状态码：400

表 5-217 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

状态码：403

表 5-218 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

请求示例

设置用户权限，为root用户授予发布和订阅topic-test的权限。

```
POST https://{endpoint}/v1/{project_id}/instances/{instance_id}/topics/accesspolicy

{
  "topics" : [ {
    "name" : "topic-test",
    "policies" : [ {
      "user_name" : "root",
      "access_policy" : "all"
    } ]
  } ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

设置用户权限，为root用户授予发布和订阅topic-test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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;
```

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

public class UpdateTopicAccessPolicySolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateTopicAccessPolicyRequest request = new UpdateTopicAccessPolicyRequest();
        request.withInstanceId("{instance_id}");
        UpdateTopicAccessPolicyReq body = new UpdateTopicAccessPolicyReq();
        List<AccessPolicyEntity> listTopicsPolicies = new ArrayList<>();
        listTopicsPolicies.add(
            new AccessPolicyEntity()
                .withUserName("root")
                .withAccessPolicy(AccessPolicyEntity.AccessPolicyEnum.fromValue("all"))
        );
        List<AccessPolicyTopicEntity> listbodyTopics = new ArrayList<>();
        listbodyTopics.add(
            new AccessPolicyTopicEntity()
                .withName("topic-test")
                .withPolicies(listTopicsPolicies)
        );
        body.withTopics(listbodyTopics);
        request.withBody(body);
        try {
            UpdateTopicAccessPolicyResponse response = client.updateTopicAccessPolicy(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

设置用户权限，为root用户授予发布和订阅topic-test的权限。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = UpdateTopicAccessPolicyRequest()
        request.instance_id = "{instance_id}"
        listPoliciesTopics = [
            AccessPolicyEntity(
                user_name="root",
                access_policy="all"
            )
        ]
        listTopicsbody = [
            AccessPolicyTopicEntity(
                name="topic-test",
                policies=listPoliciesTopics
            )
        ]
        request.body = UpdateTopicAccessPolicyReq(
            topics=listTopicsbody
        )
        response = client.update_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

设置用户权限，为root用户授予发布和订阅topic-test的权限。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateTopicAccessPolicyRequest{}
    request.InstanceId = "{instance_id}"
    userNamePolicies:= "root"
    accessPolicyPolicies:= model.GetAccessPolicyEntityAccessPolicyEnum().ALL
    var listPoliciesTopics = []model.AccessPolicyEntity{
        {
            UserName: &userNamePolicies,
            AccessPolicy: &accessPolicyPolicies,
        },
    }
    var listTopicsbody = []model.AccessPolicyTopicEntity{
        {
            Name: "topic-test",
            Policies: listPoliciesTopics,
        },
    }
    request.Body = &model.UpdateTopicAccessPolicyReq{
        Topics: listTopicsbody,
    }
    response, err := client.UpdateTopicAccessPolicy(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	更新成功。
400	参数无效。
403	鉴权失败。

错误码

请参见**错误码**。

5.8 消息管理

5.8.1 查询消息

功能介绍

查询消息的偏移量和消息内容。

先根据时间戳查询消息的偏移量，再根据偏移量查询消息内容。

调用方法

请参见**如何调用API**。

URI

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

表 5-219 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 5-220 Query 参数

参数	是否必选	参数类型	描述
topic	是	String	Topic名称。 Topic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。
asc	否	Boolean	是否按照时间排序。
start_time	否	String	开始时间。 Unix毫秒时间戳。 查询消息偏移量时，为必选参数。
end_time	否	String	结束时间。 Unix毫秒时间戳。 查询消息偏移量时，为必选参数。
limit	否	String	每一页显示的message数量。
offset	否	String	页数。
download	否	Boolean	是否下载。

参数	是否必选	参数类型	描述
message_offset	否	String	消息偏移量。 **查询消息内容时，为必选参数。** 若start_time、end_time参数不为空，该参数无效。
partition	否	String	分区。 **查询消息内容时，为必选参数。** 若start_time、end_time参数不为空，该参数无效。
keyword	否	String	关键词。 取值范围为0~50。

请求参数

无

响应参数

状态码：200

表 5-221 响应 Body 参数

参数	参数类型	描述
messages	Array of **MessagesEntity** objects	消息列表。
total	Long	消息总条数。
size	Long	每页消息条数。

表 5-222 MessagesEntity

参数	参数类型	描述
topic	String	topic名称。
partition	Integer	消息所在的分区。
key	String	消息key。
value	String	消息内容。
size	Integer	消息大小。

参数	参数类型	描述
timestamp	Long	生产消息的时间。 格式为Unix时间戳。单位为毫秒。
huge_message	Boolean	大数据标识。
message_offset	Long	消息偏移量。
message_id	String	消息ID。
app_id	String	应用ID。
tag	String	消息标签。

状态码：400

表 5-223 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

状态码：403

表 5-224 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

请求示例

- 查询消息偏移量。

  ```
  GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/messages?
  asc=false&end_time=1608609032042&limit=10&offset=0&start_time=1608608432042&topic=topic-test
  ```

- 查询消息内容。

  ```
  GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/messages?
  download=false&message_offset=0&partition=0&topic=topic-test
  ```

响应示例

状态码：200

查询成功。

```
{
  "messages" : [ {
```

```
    "topic" : "topic-test",
    "partition" : 0,
    "value" : "hello world",
    "size" : 21,
    "timestamp" : 1607598463502,
    "huge_message" : false,
    "message_offset" : 4,
    "message_id" : "",
    "app_id" : "",
    "tag" : ""
  } ],
  "total" : 1,
  "size" : 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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowInstanceMessagesSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowInstanceMessagesRequest request = new ShowInstanceMessagesRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowInstanceMessagesResponse response = client.showInstanceMessages(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowInstanceMessagesRequest()
        request.instance_id = "{instance_id}"
        response = client.show_instance_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowInstanceMessagesRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowInstanceMessages(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。
400	参数无效。
403	鉴权失败。

错误码

请参见**错误码**。

5.8.2 查询分区指定偏移量的消息

功能介绍

查询分区指定偏移量的消息。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/message

表 5-225 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	Topic名称。 Topic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。
partition	是	Integer	分区编号。

表 5-226 Query 参数

参数	是否必选	参数类型	描述
message_offset	是	String	消息位置。

请求参数

无

响应参数

状态码：200

表 5-227 响应 Body 参数

参数	参数类型	描述
message	Array of **ShowPartitionMessageEntity** objects	消息列表。

表 5-228 ShowPartitionMessageEntity

参数	参数类型	描述
key	String	消息的key。
value	String	消息内容。
topic	String	Topic名称。

参数	参数类型	描述
partition	Integer	分区编号。
message_offset	Long	消息位置。
size	Integer	消息大小，单位字节。
timestamp	Long	生产消息的时间。 格式为Unix时间戳。单位为毫秒。

请求示例

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/message?message_offset={message_offset}
```

响应示例

状态码：200

查询分区指定偏移量的消息成功。

```
{
  "message" : [ {
    "topic" : "mytest",
    "partition" : 0,
    "message_offset" : 7,
    "key" : null,
    "value" : "kasjdf",
    "size" : 6,
    "timestamp" : 1568125036045
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowPartitionMessageSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowPartitionMessageRequest request = new ShowPartitionMessageRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        request.withPartition({partition});
        try {
            ShowPartitionMessageResponse response = client.showPartitionMessage(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowPartitionMessageRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        request.partition = {partition}
        response = client.show_partition_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

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowPartitionMessageRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    request.Partition = int32({partition})
    response, err := client.ShowPartitionMessage(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	查询分区指定偏移量的消息成功。

错误码

请参见**错误码**。

5.8.3 查询分区指定时间段的消息

功能介绍

查询分区指定时间段的消息。

调用方法

请参见**如何调用API**。

URI

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

表 5-229 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	Topic名称。 Topic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。

表 5-230 Query 参数

参数	是否必选	参数类型	描述
start_time	否	String	查询起始时间，为unix时间戳格式，默认值为0。
end_time	否	String	查询结束时间，为unix时间戳格式，默认值为系统当前时间。
limit	否	Integer	单页返回消息数，默认值为10。
offset	否	Integer	偏移量，表示从此偏移量开始查询， offset大于等于0。
partition	否	String	分区编号，默认值为-1，若传入值为-1，则查询所有分区。

请求参数

无

响应参数

状态码：200

表 5-231 响应 Body 参数

参数	参数类型	描述
messages	Array of **messages** objects	消息列表。
messages_count	Integer	消息总数。
offsets_count	Integer	总页数。
offset	Integer	当前页数。

表 5-232 messages

参数	参数类型	描述
topic	String	topic名称。
partition	Integer	分区编号。
message_offset	Long	消息编号。
size	Integer	消息大小，单位字节。
timestamp	Long	生产消息的时间。 格式为Unix时间戳。单位为毫秒。

请求示例

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

响应示例

状态码：200

查询分区指定时间段的消息成功。

```
{
  "messages" : [ {
    "topic" : "mytest",
    "partition" : 0,
    "message_offset" : 7,
    "size" : 6,
    "timestamp" : 1568125036045
  } ],
  "messages_count" : 1,
```

```
  "offsets_count" : 1,
  "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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowMessagesSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowMessagesRequest request = new ShowMessagesRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        try {
            ShowMessagesResponse response = client.showMessages(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
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowMessagesRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        response = client.show_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())
```

```
    request := &model.ShowMessagesRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    response, err := client.ShowMessages(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	查询分区指定时间段的消息成功。

错误码

请参见**错误码**。

5.8.4 查询分区最早消息的位置

功能介绍

查询分区最早消息的位置。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/beginning-message

表 5-233 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。

参数	是否必选	参数类型	描述
topic	是	String	Topic名称。 Topic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。
partition	是	Integer	分区编号。

请求参数

无

响应参数

状态码：200

表 5-234 响应 Body 参数

参数	参数类型	描述
topic	String	Topic名称。
partition	Integer	分区编号。
offset	Integer	最新消息位置。
timestamp	Long	生产消息的时间。 格式为Unix时间戳。单位为毫秒。

请求示例

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/beginning-message
```

响应示例

状态码：200

查询分区最早消息的位置成功。

```
{
  "topic" : "mytest",
  "partition" : 0,
  "offset" : 9,
  "timestamp" : 1568125039164
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowPartitionBeginningMessageSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowPartitionBeginningMessageRequest request = new ShowPartitionBeginningMessageRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        request.withPartition({partition});
        try {
            ShowPartitionBeginningMessageResponse response = client.showPartitionBeginningMessage(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowPartitionBeginningMessageRequest()
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        request.partition = {partition}
        response = client.show_partition_beginning_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

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowPartitionBeginningMessageRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    request.Partition = int32({partition})
    response, err := client.ShowPartitionBeginningMessage(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	查询分区最早消息的位置成功。

错误码

请参见**错误码**。

5.8.5 查询分区最新消息的位置

功能介绍

查询分区最新消息的位置。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/end-message

表 5-235 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	Topic名称。 Topic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。
partition	是	Integer	分区编号。

请求参数

无

响应参数

状态码：200

表 5-236 响应 Body 参数

参数	参数类型	描述
topic	String	Topic名称。
partition	Integer	分区编号。
offset	Integer	最新消息位置。
timestamp	Long	生产消息的时间。 格式为Unix时间戳。单位为毫秒。

请求示例

```
GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions/{partition}/end-message
```

响应示例

状态码：200

查询分区最新消息的位置成功。

```
{
  "topic" : "mytest",
  "partition" : 0,
  "offset" : 9,
  "timestamp" : 1568125039164
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowPartitionEndMessageSolution {
```

```
    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowPartitionEndMessageRequest request = new ShowPartitionEndMessageRequest();
        request.withInstanceId("{instance_id}");
        request.withTopic("{topic}");
        request.withPartition({partition});
        try {
            ShowPartitionEndMessageResponse response = client.showPartitionEndMessage(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowPartitionEndMessageRequest()
```

```
        request.instance_id = "{instance_id}"
        request.topic = "{topic}"
        request.partition = {partition}
        response = client.show_partition_end_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

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowPartitionEndMessageRequest{}
    request.InstanceId = "{instance_id}"
    request.Topic = "{topic}"
    request.Partition = int32({partition})
    response, err := client.ShowPartitionEndMessage(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	查询分区最新消息的位置成功。

错误码

请参见**错误码**。

5.8.6 Kafka 删除消息

功能介绍

Kafka删除消息。

调用方法

请参见**如何调用API**。

URI

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

表 5-237 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	Topic名称。

请求参数

表 5-238 请求 Body 参数

参数	是否必选	参数类型	描述
partitions	否	Array of **PartitionOffsetEntity** objects	分区消费位点详情

表 5-239 PartitionOffsetEntity

参数	是否必选	参数类型	描述
partition	否	Integer	分区
offset	否	Integer	消费位点

响应参数

状态码：200

表 5-240 响应 Body 参数

参数	参数类型	描述
partitions	Array of **PartitionResp** objects	分区响应信息

表 5-241 PartitionResp

参数	参数类型	描述
partition	Integer	分区
result	String	返回结果
error_code	String	返回错误码

请求示例

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

响应示例

状态码：200

删除消息成功。

```
{
  "partitions" : [ {
    "partition" : 0,
    "result" : "success",
    "error_code" : 0
  } ]
}
```

状态码

状态码	描述
200	删除消息成功。

错误码

请参见**错误码**。

5.9 后台任务管理

5.9.1 查询实例的后台任务列表

功能介绍

查询实例的后台任务列表。

调用方法

请参见**如何调用API**。

URI

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

表 5-242 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 5-243 Query 参数

参数	是否必选	参数类型	描述
start	否	Integer	开启查询的任务编号。
limit	否	Integer	查询的任务个数。
begin_time	否	String	查询任务的最小时间，格式为YYYYMMDDHHmmss。
end_time	否	String	查询任务的最大时间，格式为YYYYMMDDHHmmss。

请求参数

无

响应参数

状态码：200

表 5-244 响应 Body 参数

参数	参数类型	描述
task_count	String	任务数量。
tasks	Array of **tasks** objects	任务列表。

表 5-245 tasks

参数	参数类型	描述
id	String	任务ID。
name	String	任务名称。
user_name	String	用户名。
user_id	String	用户ID。
params	String	任务参数。
status	String	任务状态。
created_at	String	启动时间。
updated_at	String	结束时间。

请求示例

```
'GET https://{endpoint}/v2/{project_id}/instances/{instance_id}/tasks?
start={start}&limit={limit}&begin_time={begin_time}&end_time={end_time}'
```

响应示例

状态码：200

查询实例的后台任务列表

```
{
  "task_count" : "4",
  "tasks" : [ {
    "id" : "8abfa7b372160bfd0172165864064079",
    "name" : "modifyAutoTopic",
    "user_name" : "paas_dms",
    "user_id" : "3df5acbc24a54fadb62a043c9000a307",
    "params" : "{\"old_auto_status\":true,\"new_auto_status\":false}",
```

```
    "status" : "EXECUTING",
    "created_at" : "2020-05-15T03:19:51.046Z",
    "updated_at" : "2020-05-15T03:19:51.065Z"
  }, {
    "id" : "8abfa7b372160bfd017216560af83e6e",
    "name" : "changeRetentionPolicy",
    "user_name" : "paas_dms",
    "user_id" : "3df5acbc24a54fadb62a043c9000a307",
    "params" : "{\"new_retention_policy\":\"produce_reject\",\"origin_retention_policy\":\"time_base\"}",
    "status" : "SUCCESS",
    "created_at" : "2020-05-15T03:17:17.176Z",
    "updated_at" : "2020-05-15T03:17:22.162Z"
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListBackgroundTasksSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListBackgroundTasksRequest request = new ListBackgroundTasksRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListBackgroundTasksResponse response = client.listBackgroundTasks(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListBackgroundTasksRequest()
        request.instance_id = "{instance_id}"
        response = client.list_background_tasks(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListBackgroundTasksRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListBackgroundTasks(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	查询实例的后台任务列表

错误码

请参见**错误码**。

5.9.2 查询后台任务管理中的指定记录

功能介绍

查询后台任务管理中的指定记录。

调用方法

请参见**如何调用API**。

URI

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

表 5-246 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。

参数	是否必选	参数类型	描述
task_id	是	String	任务ID。

请求参数

无

响应参数

状态码：200

表 5-247 响应 Body 参数

参数	参数类型	描述
task_count	String	任务数量。
tasks	Array of **tasks** objects	任务列表。

表 5-248 tasks

参数	参数类型	描述
id	String	任务ID。
name	String	任务名称。
user_name	String	用户名。
user_id	String	用户ID。
params	String	任务参数。
status	String	任务状态。
created_at	String	启动时间。
updated_at	String	结束时间。

请求示例

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

响应示例

状态码：200

查询成功。

```
{
  "task_count" : "1",
```

```
 "tasks" : [ {
   "id" : "8abfa7b272adc5b40172b73130065ae7",
   "name" : "bindInstancePublicIp",
   "user_name" : "paas_dms",
   "user_id" : "3df5acbc24a54fadb62a043c9000a307",
   "params" : "{\"public_ip_id\":\"1aea7aed-e7d8-40ea-b3de-6f3ee9d5db9f\",\"public_ip_address
\":\"100.93.2.18\",\"enable_public_ip\":true}",
   "status" : "SUCCESS",
   "created_at" : "2020-06-15T08:55:53.606Z",
   "updated_at" : "2020-06-15T08:55:56.600Z"
 } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowBackgroundTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowBackgroundTaskRequest request = new ShowBackgroundTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            ShowBackgroundTaskResponse response = client.showBackgroundTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowBackgroundTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.show_background_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowBackgroundTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.ShowBackgroundTask(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**。

5.9.3 删除后台任务管理中的指定记录

功能介绍

删除后台任务管理中的指定记录。

调用方法

请参见**如何调用API**。

URI

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

表 5-249 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。
task_id	是	String	任务ID。

请求参数

无

响应参数

无

请求示例

删除后台任务管理中的指定记录。

```
DELETE https://{endpoint}/v2/{project_id}/instances/{instance_id}/tasks/{task_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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class DeleteBackgroundTaskSolution {

    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);
```

```
        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteBackgroundTaskRequest request = new DeleteBackgroundTaskRequest();
        request.withInstanceId("{instance_id}");
        request.withTaskId("{task_id}");
        try {
            DeleteBackgroundTaskResponse response = client.deleteBackgroundTask(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = DeleteBackgroundTaskRequest()
        request.instance_id = "{instance_id}"
        request.task_id = "{task_id}"
        response = client.delete_background_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteBackgroundTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.TaskId = "{task_id}"
    response, err := client.DeleteBackgroundTask(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	删除后台任务成功。

错误码

请参见**错误码**。

5.10 标签管理

5.10.1 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}/kafka/{instance_id}/tags/action

表 5-250 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 5-251 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 5-252 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 _ . : = + - @ ● 不能以_sys_开头。 ● 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 _ . : = + - @

响应参数

无

请求示例

创建实例标签，标签名为key1、key2，值为value1、value2。

```
POST https://{endpoint}/v2/{project_id}/kafka/{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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class BatchCreateOrDeleteKafkaTagSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchCreateOrDeleteKafkaTagRequest request = new BatchCreateOrDeleteKafkaTagRequest();
        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 {
            BatchCreateOrDeleteKafkaTagResponse response = client.batchCreateOrDeleteKafkaTag(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

创建实例标签，标签名为key1、key2，值为value1、value2。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchCreateOrDeleteKafkaTagRequest()
        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_kafka_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
```

```
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchCreateOrDeleteKafkaTagRequest{}
    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.BatchCreateOrDeleteKafkaTag(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**。

5.10.2 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}/kafka/{instance_id}/tags

表 5-253 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 5-254 响应 Body 参数

参数	参数类型	描述
tags	Array of **TagEntity** objects	标签列表

表 5-255 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_ . : = + - @ ● 不能以_sys_开头。 ● 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_ . : = + - @

请求示例

```
GET https://{endpoint}/v2/{project_id}/kafka/{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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
```

```
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowKafkaTagsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowKafkaTagsRequest request = new ShowKafkaTagsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowKafkaTagsResponse response = client.showKafkaTags(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

```
    try:
        request = ShowKafkaTagsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_kafka_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowKafkaTagsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowKafkaTags(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**。

5.10.3 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}/kafka/tags

表 5-256 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 5-257 响应 Body 参数

参数	参数类型	描述
tags	Array of **TagMultyValueEntity** objects	标签列表

表 5-258 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

```
GET https://{endpoint}/v2/{project_id}/kafka/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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowKafkaProjectTagsSolution {

    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);
```

```
        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowKafkaProjectTagsRequest request = new ShowKafkaProjectTagsRequest();
        try {
            ShowKafkaProjectTagsResponse response = client.showKafkaProjectTags(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowKafkaProjectTagsRequest()
        response = client.show_kafka_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowKafkaProjectTagsRequest{}
    response, err := client.ShowKafkaProjectTags(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**。

5.11 诊断管理

5.11.1 消息积压诊断预检查

功能介绍

消息积压诊断预检查

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/kafka/instances/{instance_id}/diagnosis-check

表 5-259 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 5-260 Query 参数

参数	是否必选	参数类型	描述
group	是	String	消费组名称
topic	是	String	主题名称

请求参数

无

响应参数

状态码：200

表 5-261 响应 Body 参数

参数	参数类型	描述
[数组元素]	Array of **KafkaDiagnosisCheckEntity** objects	Kafka消息积压诊断预检查返回对象

表 5-262 KafkaDiagnosisCheckEntity

参数	参数类型	描述
name	String	预检查项名称
reason	String	预检查失败原因

参数	参数类型	描述
success	Boolean	预检查是否正常

状态码：400

表 5-263 响应 Body 参数

参数	参数类型	描述
error_code	String	错误码。
error_msg	String	错误描述。

请求示例

对消费组group-1所订阅的topic-1进行消息积压诊断预检查

```
GET https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/diagnosis-check?
group=group-1&topic=topic-1
```

响应示例

状态码：200

消息积压诊断预检查成功

```
[ {
  "name" : "RAM",
  "success" : false,
  "reason" : "1,2"
}, {
  "name" : "CPU",
  "success" : false,
  "reason" : "1,2"
}, {
  "name" : "SUBSCRIPTION",
  "success" : false,
  "reason" : "0,2"
}, {
  "name" : "LAG",
  "success" : false,
  "reason" : "{\"0\":10159908,\"1\":9768464,\"2\":9361706}"
}, {
  "name" : "PRODUCE",
  "success" : false,
  "reason" : "{\"1694571420000\":14.159}"
} ]
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowDiagnosisPreCheckSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowDiagnosisPreCheckRequest request = new ShowDiagnosisPreCheckRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowDiagnosisPreCheckResponse response = client.showDiagnosisPreCheck(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"
```

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

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

    try:
        request = ShowDiagnosisPreCheckRequest()
        request.instance_id = "{instance_id}"
        response = client.show_diagnosis_pre_check(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowDiagnosisPreCheckRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowDiagnosisPreCheck(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	消息积压诊断预检查成功
400	参数无效。

错误码

请参见**错误码**。

5.11.2 创建消息积压诊断任务

功能介绍

创建消息积压诊断任务

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks

表 5-264 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 5-265 请求 Body 参数

参数	是否必选	参数类型	描述
group_name	是	String	消费组名称
topic_name	是	String	topic名称

响应参数

状态码：200

表 5-266 响应 Body 参数

参数	参数类型	描述
report_id	String	诊断报告ID。

请求示例

创建消费组group-test，主题topic-test的消息积压诊断任务

```
POST https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks

{
  "group_name" : "group-test",
  "topic_name" : "topic-test"
}
```

响应示例

状态码：200

消息积压诊断任务ID

```
{
  "report_id" : "0e35a9f4-e75d-4fd6-b230-280860f666f7"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建消费组group-test，主题topic-test的消息积压诊断任务

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateMessageDiagnosisTaskSolution {

    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);

    KafkaClient client = KafkaClient.newBuilder()
            .withCredential(auth)
            .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
            .build();
    CreateMessageDiagnosisTaskRequest request = new CreateMessageDiagnosisTaskRequest();
    request.withInstanceId("{instance_id}");
    CreateMessageDiagnosisTaskReq body = new CreateMessageDiagnosisTaskReq();
    body.withTopicName("topic-test");
    body.withGroupName("group-test");
    request.withBody(body);
    try {
        CreateMessageDiagnosisTaskResponse response = client.createMessageDiagnosisTask(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

创建消费组group-test，主题topic-test的消息积压诊断任务

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = CreateMessageDiagnosisTaskRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateMessageDiagnosisTaskReq(
            topic_name="topic-test",
            group_name="group-test"
        )
        response = client.create_message_diagnosis_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

创建消费组group-test，主题topic-test的消息积压诊断任务

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateMessageDiagnosisTaskRequest{}
    request.InstanceId = "{instance_id}"
    request.Body = &model.CreateMessageDiagnosisTaskReq{
        TopicName: "topic-test",
        GroupName: "group-test",
    }
    response, err := client.CreateMessageDiagnosisTask(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	消息积压诊断任务ID

错误码

请参见**错误码**。

5.11.3 查询消息积压诊断报告列表

功能介绍

查询消息积压诊断报告列表

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks

表 5-267 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 5-268 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	偏移量，表示查询该偏移量后面的记录
limit	否	Integer	查询返回记录的数量限制

请求参数

无

响应参数

状态码：200

表 5-269 响应 Body 参数

参数	参数类型	描述
report_list	Array of **KafkaMessageDiagnosisReportInfoEntity** objects	诊断报告列表
total_num	Integer	诊断报告总数

表 5-270 KafkaMessageDiagnosisReportInfoEntity

参数	参数类型	描述
report_id	String	诊断报告ID
status	String	消息积压诊断任务状态。 • diagnosing：诊断中 • failed：诊断失败 • deleted：手动删除 • finished：诊断完成 • normal：诊断结果正常 • abnormal：诊断结果异常
begin_time	String	诊断任务开始时间
end_time	String	诊断任务结束时间
group_name	String	该次诊断任务诊断的消费组名称
topic_name	String	该次诊断任务诊断的topic名称
accumulated_partitions	Integer	该次诊断任务发现的存在消息堆积的分区数

请求示例

查询消息积压诊断报告列表。

```
GET https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks?offset=1&limit=10
```

响应示例

状态码：200

查询消息积压诊断报告列表成功

```
{
  "report_list" : [ {
    "report_id" : "89b202d5-1f34-4a89-af9d-698496d0b7b1",
    "status" : "diagnosing",
```

```
  "begin_time" : "2022-11-11T10:00:00.000Z",
  "end_time" : "2022-11-11T10:00:00.000Z",
  "group_name" : "group-test",
  "topic_name" : "topic-test",
  "accumulated_partitions" : 3
 } ],
 "total_num" : 1
}
```

状态码

状态码	描述
200	查询消息积压诊断报告列表成功

错误码

请参见**错误码**。

5.11.4 批量删除消息积压诊断报告

功能介绍

批量删除消息积压诊断报告

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks

表 5-271 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 5-272 请求 Body 参数

参数	是否必选	参数类型	描述
report_id_list	是	Array of strings	待删除report id列表

响应参数

状态码：200

表 5-273 响应 Body 参数

参数	参数类型	描述
results	Array of **results** objects	诊断报告删除结果

表 5-274 results

参数	参数类型	描述
result	Boolean	报告删除结果
id	String	报告ID

请求示例

批量删除消息积压诊断报告。

```
DELETE https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis-tasks

{
  "report_id_list" : [ "0e35a9f4-e75d-4fd6-b230-280860f666f7" ]
}
```

响应示例

状态码：200

批量删除消息积压诊断报告成功

```
{
  "results" : [ {
    "result" : "true",
    "id" : "0e35a9f4-e75d-4fd6-b230-280860f666f7"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

批量删除消息积压诊断报告。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;

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

public class BatchDeleteMessageDiagnosisReportsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        BatchDeleteMessageDiagnosisReportsRequest request = new BatchDeleteMessageDiagnosisReportsRequest();
        request.withInstanceId("{instance_id}");
        BatchDeleteMessageDiagnosisReportsReq body = new BatchDeleteMessageDiagnosisReportsReq();
        List<String> listbodyReportIdList = new ArrayList<>();
        listbodyReportIdList.add("0e35a9f4-e75d-4fd6-b230-280860f666f7");
        body.withReportIdList(listbodyReportIdList);
        request.withBody(body);
        try {
            BatchDeleteMessageDiagnosisReportsResponse response = client.batchDeleteMessageDiagnosisReports(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

批量删除消息积压诊断报告。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = BatchDeleteMessageDiagnosisReportsRequest()
        request.instance_id = "{instance_id}"
        listReportIdListbody = [
            "0e35a9f4-e75d-4fd6-b230-280860f666f7"
        ]
        request.body = BatchDeleteMessageDiagnosisReportsReq(
            report_id_list=listReportIdListbody
        )
        response = client.batch_delete_message_diagnosis_reports(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

批量删除消息积压诊断报告。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

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

```
    request.InstanceId = "{instance_id}"
    var listReportIdListbody = []string{
        "0e35a9f4-e75d-4fd6-b230-280860f666f7",
    }
    request.Body = &model.BatchDeleteMessageDiagnosisReportsReq{
        ReportIdList: listReportIdListbody,
    }
    response, err := client.BatchDeleteMessageDiagnosisReports(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	批量删除消息积压诊断报告成功

错误码

请参见**错误码**。

5.11.5 查询诊断报告详情

功能介绍

查询诊断报告详情

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis/{report_id}

表 5-275 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
report_id	是	String	消息积压诊断报告ID

请求参数

无

响应参数

状态码：200

表 5-276 响应 Body 参数

参数	参数类型	描述
abnormal_item_num	Integer	诊断异常的诊断项总和
failed_item_num	Integer	诊断失败的诊断项总和
normal_item_num	Integer	诊断正常的诊断项总和
diagnosis_dimension_list	Array of **KafkaMessageDiagnosisDimensionEntity** objects	诊断维度列表

表 5-277 KafkaMessageDiagnosisDimensionEntity

参数	参数类型	描述
name	String	诊断维度名称
abnormal_num	Integer	该诊断维度下，异常的诊断项总数
failed_num	Integer	该诊断维度下，诊断失败的诊断项总和
diagnosis_item_list	Array of **KafkaMessageDiagnosisItemEntity** objects	诊断项列表

表 5-278 KafkaMessageDiagnosisItemEntity

参数	参数类型	描述
name	String	诊断项名称
result	String	诊断结果
cause_ids	Array of **KafkaMessageDiagnosisConclusionEntity** objects	诊断异常原因列表

参数	参数类型	描述
advice_ids	Array of KafkaMessageDiagnosisConclusionEntity objects	诊断异常建议列表
partitions	Array of integers	诊断异常受影响的分区列表
failed_partitions	Array of integers	诊断失败的分区列表
broker_ids	Array of integers	诊断异常受影响的broker列表

表 5-279 KafkaMessageDiagnosisConclusionEntity

参数	参数类型	描述
id	Integer	诊断结论ID
params	Map<String,String>	诊断结论参数列表

请求示例

查询诊断报告详情。

```
GET https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/message-diagnosis/{report_id}
```

响应示例

状态码：200

查询消息积压诊断报告详情成功

```
{
  "abnormal_item_num" : 2,
  "failed_item_num" : 2,
  "normal_item_num" : 1,
  "diagnosis_dimension_list" : [ {
    "name" : "server",
    "abnormal_num" : 1,
    "failed_num" : 1,
    "diagnosis_item_list" : [ {
      "name" : "fetch_error",
      "result" : "abnormal",
      "cause_ids" : [ {
        "id" : 1,
        "params" : {
          "THRESHOLD" : "90",
          "ERROR_CODE" : "[1,2,3]"
        }
      } ],
      "advice_ids" : [ {
        "id" : 1,
        "params" : { }
      } ],
      "partitions" : [ 1, 2, 3 ],
      "failed_partitions" : [ 4, 5, 6 ],
```

```
    "broker_ids" : [ 1, 2, 3 ]
   } ]
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowMessageDiagnosisReportSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowMessageDiagnosisReportRequest request = new ShowMessageDiagnosisReportRequest();
        request.withInstanceId("{instance_id}");
        request.withReportId("{report_id}");
        try {
            ShowMessageDiagnosisReportResponse response = client.showMessageDiagnosisReport(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
```

```
import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowMessageDiagnosisReportRequest()
        request.instance_id = "{instance_id}"
        request.report_id = "{report_id}"
        response = client.show_message_diagnosis_report(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())
```

```
    request := &model.ShowMessageDiagnosisReportRequest{}
    request.InstanceId = "{instance_id}"
    request.ReportId = "{report_id}"
    response, err := client.ShowMessageDiagnosisReport(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	查询消息积压诊断报告详情成功

错误码

请参见**错误码**。

5.12 其他接口

5.12.1 查询维护时间窗时间段

功能介绍

查询维护时间窗开始时间和结束时间。

调用方法

请参见**如何调用API**。

URI

GET /v2/instances/maintain-windows

请求参数

无

响应参数

状态码：200

表 5-280 响应 Body 参数

参数	参数类型	描述
maintain_windows	Array of **MaintainWindowsEntity** objects	支持的维护时间窗列表。

表 5-281 MaintainWindowsEntity

参数	参数类型	描述
default	Boolean	是否为默认时间段。
end	String	维护时间窗结束时间。
begin	String	维护时间窗开始时间。
seq	Integer	序号。

请求示例

```
GET https://{endpoint}/v2/instances/maintain-windows
```

响应示例

状态码：200

查询成功。

```
{
  "maintain_windows" : [ {
    "default" : false,
    "seq" : 1,
    "begin" : "22",
    "end" : "02"
  }, {
    "default" : true,
    "seq" : 2,
    "begin" : "02",
    "end" : "06"
  }, {
    "default" : false,
    "seq" : 3,
    "begin" : "06",
    "end" : "10"
  }, {
    "default" : false,
    "seq" : 4,
    "begin" : "10",
    "end" : "14"
  }, {
    "default" : false,
    "seq" : 5,
    "begin" : "14",
    "end" : "18"
  }, {
    "default" : false,
    "seq" : 6,
    "begin" : "18",
```

```
    "end" : "22"
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowMaintainWindowsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowMaintainWindowsRequest request = new ShowMaintainWindowsRequest();
        try {
            ShowMaintainWindowsResponse response = client.showMaintainWindows(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
```

```
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]

    credentials = BasicCredentials(ak, sk)

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

    try:
        request = ShowMaintainWindowsRequest()
        response = client.show_maintain_windows(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowMaintainWindowsRequest{}
    response, err := client.ShowMaintainWindows(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**。

5.12.2 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 5-282 响应 Body 参数

参数	参数类型	描述
region_id	String	区域ID。
available_zones	Array of **AvailableZonesResp** objects	可用区数组。

表 5-283 AvailableZonesResp

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	可用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

```
GET https://{endpoint}/v2/available-zones
```

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" : "d539378ec1314c85b76fefa3f7071458",
    "code" : "xxx",
    "name" : "可用区2",
    "port" : "8003",
    "resource_availability" : "true",
    "default_az" : true,
    "remain_time" : 9223372036854776000,
    "ipv6_enable" : false
  }, {
    "soldOut" : false,
    "id" : "9f1c5806706d4c1fb0eb72f0a9b18c77",
    "code" : "xxx",
    "name" : "可用区3",
    "port" : "443",
    "resource_availability" : "true",
    "default_az" : false,
    "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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
```

```
    credentials = BasicCredentials(ak, sk)

    client = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            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**。

5.12.3 查询产品规格列表

功能介绍

查询产品规格列表。

调用方法

请参见**如何调用API**。

URI

GET /v2/{engine}/products

表 5-284 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。

表 5-285 Query 参数

参数	是否必选	参数类型	描述
product_id	否	String	产品ID。

请求参数

无

响应参数

状态码：200

表 5-286 响应 Body 参数

参数	参数类型	描述
engine	String	分布式消息服务的产品类型。
versions	Array of strings	支持的产品版本类型。
products	Array of **ListEngineProductsEntity** objects	产品规格的详细信息。

表 5-287 ListEngineProductsEntity

参数	参数类型	描述
type	String	产品类型。当前产品类型有单机和集群。
product_id	String	产品ID。
ecs_flavor_id	String	底层资源类型。
billing_code	String	账单计费类型。
arch_types	Array of strings	CPU架构。
charging_mode	Array of strings	计费模式。monthly：包年/包月类型。hourly：按需类型。
ios	Array of **ListEngineIosEntity** objects	支持的磁盘IO类型列表。
support_features	Array of **ListEngineSupportFeaturesEntity** objects	当前规格实例支持的功能特性列表。
properties	**ListEnginePropertiesEntity** object	当前规格实例的属性。

表 5-288 ListEngineIosEntity

参数	参数类型	描述
io_spec	String	磁盘IO编码。
type	String	磁盘类型。
available_zones	Array of strings	可用区。
unavailable_zones	Array of strings	不可用区。

表 5-289 ListEngineSupportFeaturesEntity

参数	参数类型	描述
name	String	功能名称。
properties	**ListEngineSupportFeaturesPropertiesEntity** object	实例支持的功能属性描述。

表 5-290 ListEngineSupportFeaturesPropertiesEntity

参数	参数类型	描述
max_task	String	转储功能的最大任务数。
min_task	String	转储功能的最小任务数。
max_node	String	转储功能的最大节点数。
min_node	String	转储功能的最小节点数。

表 5-291 ListEnginePropertiesEntity

参数	参数类型	描述
max_partition_per_broker	String	每个Broker的最大分区数。
max_broker	String	Broker的最大个数。
max_storage_per_node	String	每个节点的最大存储。单位为GB。
max_consumer_per_broker	String	每个Broker的最大消费者数。
min_broker	String	Broker的最小个数。
max_bandwidth_per_broker	String	每个Broker的最大带宽。
min_storage_per_node	String	每个节点的最小存储。单位为GB。
max_tps_per_broker	String	每个Broker的最大TPS。
product_alias	String	product_id的别名。

请求示例

```
GET https://{endpoint}/v2/kafka/products
```

响应示例

状态码：200

查询产品规格列表成功。


```
{
  "engine" : "kafka",
  "versions" : [ "1.1.0", "2.3.0" ],
  "products" : [ {
    "type" : "cluster",
    "product_id" : "c6.2u4g.cluster",
    "ecs_flavor_id" : "c6.large.2",
    "billing_code" : "dms.platinum.c6",
    "arch_types" : [ "X86" ],
    "charging_mode" : [ "monthly", "hourly" ],
    "ios" : [ {
      "io_spec" : "dms.physical.storage.high.v2",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
        "max_node" : "10",
        "min_task" : "1",
        "min_node" : "2"
      }
    } ],
    "properties" : {
      "max_partition_per_broker" : "250",
      "max_broker" : "30",
      "max_storage_per_node" : "10000",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "3",
      "max_bandwidth_per_broker" : "100",
      "min_storage_per_node" : "200",
      "max_tps_per_broker" : "30000",
      "product_alias" : "kafka.2u4g.cluster"
    }
  }, {
    "type" : "cluster",
    "product_id" : "c6.4u8g.cluster",
    "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",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
```

```
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "500",
    "max_broker" : "30",
    "max_storage_per_node" : "20000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "400",
    "max_tps_per_broker" : "100000",
    "product_alias" : "kafka.4u8g.cluster"
  }
}, {
  "type" : "cluster",
  "product_id" : "c6.8u16g.cluster",
  "ecs_flavor_id" : "c6.2xlarge.2",
  "billing_code" : "dms.platinum.c6",
  "arch_types" : [ "X86" ],
  "charging_mode" : [ "monthly", "hourly" ],
  "ios" : [ {
    "io_spec" : "dms.physical.storage.high.v2",
    "type" : "evs",
    "available_zones" : [ "xxx", "xxx" ],
    "unavailable_zones" : [ "xxx", "xxx" ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "type" : "evs",
    "available_zones" : [ "xxx", "xxx" ],
    "unavailable_zones" : [ "xxx", "xxx" ]
  } ],
  "support_features" : [ {
    "name" : "connector_obs",
    "properties" : {
      "max_task" : "10",
      "max_node" : "10",
      "min_task" : "1",
      "min_node" : "2"
    }
  } ],
  "properties" : {
    "max_partition_per_broker" : "1000",
    "max_broker" : "30",
    "max_storage_per_node" : "30000",
    "max_consumer_per_broker" : "4000",
    "min_broker" : "3",
    "max_bandwidth_per_broker" : "100",
    "min_storage_per_node" : "800",
    "max_tps_per_broker" : "150000",
    "product_alias" : "kafka.8u16g.cluster"
  }
}, {
  "type" : "cluster",
  "product_id" : "c6.12u24g.cluster",
  "ecs_flavor_id" : "c6.3xlarge.2",
  "billing_code" : "dms.platinum.c6",
  "arch_types" : [ "X86" ],
  "charging_mode" : [ "monthly", "hourly" ],
  "ios" : [ {
    "io_spec" : "dms.physical.storage.high.v2",
    "type" : "evs",
    "available_zones" : [ "xxx", "xxx" ],
    "unavailable_zones" : [ "xxx", "xxx" ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "type" : "evs",
```

```
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
        "max_node" : "10",
        "min_task" : "1",
        "min_node" : "2"
      }
    } ],
    "properties" : {
      "max_partition_per_broker" : "1500",
      "max_broker" : "30",
      "max_storage_per_node" : "30000",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "3",
      "max_bandwidth_per_broker" : "100",
      "min_storage_per_node" : "1200",
      "max_tps_per_broker" : "200000",
      "product_alias" : "kafka.12u24g.cluster"
    }
  }, {
    "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",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    }, {
      "io_spec" : "dms.physical.storage.ultra.v2",
      "type" : "evs",
      "available_zones" : [ "xxx", "xxx" ],
      "unavailable_zones" : [ "xxx", "xxx" ]
    } ],
    "support_features" : [ {
      "name" : "connector_obs",
      "properties" : {
        "max_task" : "10",
        "max_node" : "10",
        "min_task" : "1",
        "min_node" : "2"
      }
    } ],
    "properties" : {
      "max_partition_per_broker" : "2000",
      "max_broker" : "30",
      "max_storage_per_node" : "30000",
      "max_consumer_per_broker" : "4000",
      "min_broker" : "3",
      "max_bandwidth_per_broker" : "100",
      "min_storage_per_node" : "1600",
      "max_tps_per_broker" : "250000",
      "product_alias" : "kafka.16u32g.cluster"
    }
  } ]
}
```


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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListEngineProductsSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListEngineProductsRequest request = new ListEngineProductsRequest();
        request.withEngine(ListEngineProductsRequest.EngineEnum.fromValue("{engine}"));
        try {
            ListEngineProductsResponse response = client.listEngineProducts(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
```

```
    sk = os.environ["CLOUD_SDK_SK"]

    credentials = BasicCredentials(ak, sk)

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

    try:
        request = ListEngineProductsRequest()
        request.engine = "{engine}"
        response = client.list_engine_products(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListEngineProductsRequest{}
    request.Engine = model.GetListEngineProductsRequestEngineEnum().ENGINE
    response, err := client.ListEngineProducts(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	查询产品规格列表成功。

错误码

请参见**错误码**。

5.12.4 查询实例在 CES 的监控层级关系

功能介绍

查询实例在CES的监控层级关系。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/instances/{instance_id}/ces-hierarchy

表 5-292 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 5-293 响应 Body 参数

参数	参数类型	描述
dimensions	Array of **dimensions** objects	监控维度。

参数	参数类型	描述
instance_ids	Array of **instance_ids** objects	实例信息。
nodes	Array of **nodes** objects	节点信息。
queues	Array of **queues** objects	队列信息。
groups	Array of **groups** objects	消费组信息。

表 5-294 dimensions

参数	参数类型	描述
name	String	监控维度名称。
metrics	Array of strings	监控指标名称。
key_name	Array of strings	监控查询使用的key。
dim_router	Array of strings	监控维度路由。
children	Array of **children** objects	子维度列表。

表 5-295 children

参数	参数类型	描述
name	String	子维度名称。
metrics	Array of strings	监控指标名称列表。
key_name	Array of strings	监控查询使用的key。
dim_router	Array of strings	监控维度路由。

表 5-296 instance_ids

参数	参数类型	描述
name	String	实例ID。

表 5-297 nodes

参数	参数类型	描述
name	String	节点名称。

表 5-298 queues

参数	参数类型	描述
name	String	topic名称。
partitions	Array of **partitions** objects	分区列表。

表 5-299 partitions

参数	参数类型	描述
name	String	分区名称。

表 5-300 groups

参数	参数类型	描述
name	String	消费组名称。
queues	Array of **queues** objects	topic信息。

表 5-301 queues

参数	参数类型	描述
name	String	topic名称。
partitions	Array of **partitions** objects	分区信息。

表 5-302 partitions

参数	参数类型	描述
name	String	分区名称。

请求示例

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

响应示例

状态码：200

查询成功。


```
{
  "dimensions" : [ {
    "name" : "kafka_instance_id",
    "metrics" : [ "current_partitions", "current_topics", "group_messages" ],
    "key_name" : [ "instance_ids" ],
    "dim_router" : [ "kafka_instance_id" ]
  }, {
    "name" : "kafka_broker",
    "metrics" : [ "broker_data_size", "broker_messages_in_rate", "broker_bytes_out_rate",
"broker_bytes_in_rate", "broker_produce_mean", "broker_fetch_mean" ],
    "key_name" : [ "nodes" ],
    "dim_router" : [ "kafka_instance_id", "kafka_broker" ]
  }, {
    "name" : "kafka_rest",
    "metrics" : [ "rest_produce_success", "rest_produce_failed", "rest_produce_latency",
"rest_produce_msg_num", "rest_produce_flow", "rest_consume_success", "rest_consume_failed",
"rest_consume_latency", "rest_consume_msg_num", "rest_consume_flow", "rest_commit_success",
"rest_commit_failed", "rest_commit_latency", "rest_commit_msg_num", "rest_commit_flow" ],
    "key_name" : [ "nodes" ],
    "dim_router" : [ "kafka_instance_id", "kafka_rest" ]
  }, {
    "name" : "kafka_topics",
    "metrics" : [ "topic_data_size", "topic_messages_in_rate", "topic_bytes_out_rate", "topic_bytes_in_rate",
"topic_messages" ],
    "key_name" : [ "queues" ],
    "dim_router" : [ "kafka_instance_id", "kafka_topics" ],
    "children" : [ {
      "name" : "kafka_partitions",
      "metrics" : [ "produced_messages", "partition_messages" ],
      "key_name" : [ "queues", "partitions" ],
      "dim_router" : [ "kafka_instance_id", "kafka_topics", "kafka_partitions" ]
    } ]
  }, {
    "name" : "kafka_groups_partitions",
    "metrics" : [ "messages_consumed", "messages_remained" ],
    "key_name" : [ "groups", "queues", "partitions" ],
    "dim_router" : [ "kafka_instance_id", "kafka_groups", "kafka_groups_topics", "kafka_groups_partitions" ]
  } ],
  "instance_ids" : [ {
    "name" : "68f3f6a0-3741-453b-bda9-a6ff6b5bb6f7"
  } ],
  "nodes" : [ {
    "name" : "broker-0"
  }, {
    "name" : "broker-1"
  }, {
    "name" : "broker-2"
  } ],
  "queues" : [ {
    "name" : "aaaa",
    "partitions" : [ {
      "name" : "0"
    } ]
  }, {
    "name" : "mytest",
    "partitions" : [ {
      "name" : "0"
    }, {
      "name" : "1"
```

```
    }, {
      "name" : "2"
    } ]
  }, {
    "name" : "topic-84234378",
    "partitions" : [ {
      "name" : "0"
    }, {
      "name" : "1"
    }, {
      "name" : "2"
    } ]
  } ],
  "groups" : [ {
    "name" : "test-consumer-group",
    "queues" : [ {
      "name" : "mytest",
      "partitions" : [ {
        "name" : "0"
      }, {
        "name" : "1"
      }, {
        "name" : "2"
      } ]
    } ]
  } ]
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowCesHierarchySolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowCesHierarchyRequest request = new ShowCesHierarchyRequest();
```

```
        request.withInstanceId("{instance_id}");
        try {
            ShowCesHierarchyResponse response = client.showCesHierarchy(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

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.environ["CLOUD_SDK_AK"]
    sk = os.environ["CLOUD_SDK_SK"]
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ShowCesHierarchyRequest()
        request.instance_id = "{instance_id}"
        response = client.show_ces_hierarchy(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowCesHierarchyRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ShowCesHierarchy(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**。

5.12.5 查询 Kafka 产品规格核数

功能介绍

查询Kafka产品规格核数。

调用方法

请参见**如何调用API**。

URI

GET /v2/kafka/products/cores

表 5-303 Query 参数

参数	是否必选	参数类型	描述
instance_id	是	String	实例ID。
product_id	是	String	产品ID。

请求参数

无

响应参数

状态码：200

表 5-304 响应 Body 参数

参数	参数类型	描述
core_num	Integer	核数

请求示例

```
GET https://{endpoint}/v2/kafka/products/cores
```

响应示例

状态码：200

查询成功。

```
{
  "core_num" : 100
}
```

状态码

状态码	描述
200	查询成功。

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Kafka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Kafka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。

DMS for Kafka系统策略说明请参考**权限管理**。

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Kafka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。管理员可以在授权项列表中查看授权项是否支持IAM项目或企业项目，“√”表示支持，“×”表示暂不支持。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 Kafka的支持自定义策略授权项如下所示：

- **生命周期管理**，包含Kafka实例所有生命周期接口对应的授权项，如创建实例、查询实例列表、修改实例信息、批量重启或删除实例等接口。
- **实例管理**，包括Kafka实例管理接口对应的授权项，如重置密码、开启或关闭实例自动创建Topic功能等接口。
- **Smart Connect**，包括Smart Connect接口对应的授权项，如开启Smart Connect、关闭Smart Connect、创建Smart Connect任务等接口。
- **规格变更管理**，包括实例规格变更管理接口对应的授权项，如实例规格变更接口。
- **主题管理**，包括Topic管理接口对应的授权项，如创建Topic、查询Topic、修改Topic等接口。
- **用户管理**，包括用户管理接口对应的授权项，如创建用户、查询用户列表、设置用户权限等接口。
- **消息查询**，包括消息查询接口对应的授权项，如查询消息接口。
- **后台任务管理**，包括实例的后台任务管理接口对应的授权项，如查询实例的后台任务列表、查询后台任务管理中的指定记录等接口。
- **标签管理**，包括实例的标签管理接口对应的授权项，如查询实例标签、查询项目标签等接口。

生命周期管理

表 6-1 生命周期管理

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
创建实例	POST /v2/{engine}/{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	√	√

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
批量重启或删除实例	POST /v2/{project_id}/instances/action	重启：dms:instance:modifyStatus 删除：dms:instance:delete	√	√

实例管理

表 6-2 实例管理

权限	对应API接口	授权项（Action）	IAM项目(Project)	企业项目(Enterprise Project)
重置密码	POST /v2/{project_id}/instances/{instance_id}/password	dms:instance:resetAuthInfo	√	√
重置Manager密码	PUT /v2/{project_id}/instances/{instance_id}/kafka-manager-password	dms:instance:resetAuthInfo	√	√
重启Manager	PUT /v2/{project_id}/instances/{instance_id}/restart-kafka-manager	dms:instance:modifyStatus	√	√
开启或关闭实例自动创建topic功能	POST /v2/{project_id}/instances/{instance_id}/autotopic	dms:instance:modify	√	√
修改实例跨VPC访问的内网IP	POST /v2/{project_id}/instances/{instance_id}/crossvpc/modify	dms:instance:modify	√	√
重置消费组消费进度到指定位置	POST /v2/{project_id}/instances/{instance_id}/management/groups/{group}/reset-message-offset	dms:instance:modify	√	√

Smart Connect

表 6-3 Smart Connect

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
开启Smart Connect	POST /v2/{project_id}/instances/{instance_id}/connector	dms:instance:connector	√	√
关闭Smart Connect	POST /v2/{project_id}/kafka/instances/{instance_id}/delete-connector	dms:instance:connector	√	√
创建Smart Connect任务	POST /v2/{project_id}/instances/{instance_id}/connector/tasks	dms:instance:createConnectorSinkTask	√	√
查询Smart Connect任务列表	GET /v2/{project_id}/instances/{instance_id}/connector/tasks	dms:instance:listConnectorSinkTask	√	√
查询Smart Connector任务详情	GET /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}	dms:instance:getConnectorSinkTask	√	√
删除Smart Connector任务	DELETE /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}	dms:instance:deleteConnectorSinkTask	√	√
暂停Smart Connect任务	PUT /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/pause	dms:instance:updateConnectorTask	√	√
启动已暂停的Smart Connect任务	PUT /v2/{project_id}/instances/{instance_id}/connector/tasks/{task_id}/resume	dms:instance:updateConnectorTask	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
启动未启动的Smart Connect任务/重启已暂停或者运行中的Smart Connect任务	PUT /v2/{project_id}/kafka/instances/{instance_id}/connector/tasks/{task_id}/restart	dms:instance:updateConnectorTask	√	√

规格变更管理

表 6-4 规格变更管理

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
实例规格变更	POST /v2/{engine}/{project_id}/instances/{instance_id}/extend	dms:instance:scale	√	√

主题管理

表 6-5 主题管理

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
Kafka实例创建Topic	POST /v2/{project_id}/instances/{instance_id}/topics	dms:instance:modify	√	√
Kafka实例查询Topic	GET /v2/{project_id}/instances/{instance_id}/topics	dms:instance:get	√	√
修改Kafka实例Topic	PUT /v2/{project_id}/instances/{instance_id}/topics	dms:instance:modify	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
Kafka实例批量删除Topic	POST /v2/{project_id}/instances/{instance_id}/topics/delete	dms:instance:modify	√	√

用户管理

表 6-6 用户管理

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询用户列表	GET /v2/{project_id}/instances/{instance_id}/users	dms:instance:get	√	√
创建用户	POST /v2/{project_id}/instances/{instance_id}/users	dms:instance:modify	√	√
批量删除用户	PUT /v2/{project_id}/instances/{instance_id}/users	dms:instance:modify	√	√
重置用户密码	PUT /v2/{project_id}/instances/{instance_id}/users/{user_name}	dms:instance:get	√	√
查询用户权限	GET /v1/{project_id}/instances/{instance_id}/topics/{topic_name}/accesspolicy	dms:instance:get	√	√
设置用户权限	POST /v1/{project_id}/instances/{instance_id}/topics/accesspolicy	dms:instance:modify	√	√

消息查询

表 6-7 消息查询

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/{project_id}/instances/{instance_id}/messages	dms:instance:get	√	√

后台任务管理

表 6-8 后台任务管理

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询实例的后台任务列表	GET /v2/{project_id}/instances/{instance_id}/tasks	dms:instance:getBackgroundTask	√	√
查询后台任务管理中的指定记录	GET /v2/{project_id}/instances/{instance_id}/tasks/{task_id}	dms:instance:getBackgroundTask	√	√
删除后台任务管理中的指定记录	DELETE /v2/{project_id}/instances/{instance_id}/tasks/{task_id}	dms:instance:deleteBackgroundTask	√	√

标签管理

表 6-9 标签管理

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
批量添加或删除实例标签	POST /v2/{project_id}/kafka/{instance_id}/tags/action	dms:instance:modify	√	√

权限	对应API接口	授权项	IAM项目(Project)	企业项目(Enterprise Project)
查询实例标签	GET /v2/{project_id}/kafka/{instance_id}/tags	dms:instance:get	√	√
查询项目标签	GET /v2/{project_id}/kafka/tags	dms:instance:get	√	√

7 历史 API

7.1 API V1

7.1.1 实例管理类接口

7.1.1.1 创建 Kafka 实例

说明

当前页面API为历史版本API，未来可能停止维护。请使用**创建实例**。

功能介绍

创建实例，该接口创建的实例为**按需计费**的方式。

URI

POST /v1.0/{project_id}/instances

参数说明见**表7-1**。

表 7-1 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。

请求消息

请求参数

参数说明见**表7-2**。

表 7-2 参数说明

参数	类型	是否必选	说明
name	String	是	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	String	否	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	String	是	消息引擎。取值填写为：kafka。
engine_version	String	是	消息引擎的版本。
specification	String	是	Kafka实例的基准带宽，表示单位时间内传送的最大数据量，单位MB。 取值范围为： ● 100MB ● 300MB ● 600MB ● 1200MB
storage_space	Integer	是	消息存储空间，单位GB。 ● Kafka实例规格为100MB时，存储空间取值范围600GB ~ 90000GB。 ● Kafka实例规格为300MB时，存储空间取值范围1200GB ~ 90000GB。 ● Kafka实例规格为600MB时，存储空间取值范围2400GB ~ 90000GB。 ● Kafka实例规格为1200MB时，存储空间取值范围4800GB ~ 90000GB。
partition_num	Integer	是	Kafka实例的最大分区数量。 ● 参数specification为100MB时，取值300 ● 参数specification为300MB时，取值900 ● 参数specification为600MB时，取值1800 ● 参数specification为1200MB时，取值1800
access_user	String	否	当ssl_enable为true时，该参数必选，ssl_enable为false时，该参数无效。 认证用户名，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。

参数	类型	是否必选	说明
password	String	否	当ssl_enable为true时，该参数必选，ssl_enable为false时，该参数无效。 实例的认证密码。 复杂度要求： ● 输入长度为8到32位的字符串。 ● 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（`~!@#$%^&*()-_=+|[{}]:'",<.>/?）
vpc_id	String	是	虚拟私有云ID。 获取方法如下： ● 方法1：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。 ● 方法2：通过虚拟私有云服务的API接口查询，具体操作可参考**查询VPC列表**。
security_group_id	String	是	指定实例所属的安全组。 获取方法如下： ● 方法1：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。 ● 方法2：通过虚拟私有云服务的API接口查询，具体操作可参考**查询安全组列表**。
subnet_id	String	是	子网信息。 获取方法如下： ● 方法1：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。 ● 方法2：通过虚拟私有云服务的API接口查询，具体操作可参考**查询子网列表**。

参数	类型	是否必选	说明
available_zones	Array	是	创建节点到指定且有资源的可用区ID。该参数不能为空数组或者数组的值为空，详情请参考**查询可用区信息**查询得到。在查询时，请注意查看该可用区是否有资源。 创建Kafka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。参数设置参考如下示例。 ● 一个可用区："available_zones": ["a0865121f83b41cbafce65930a22a6e8"] ● 三个及以上可用区："available_zones": ["a0865121f83b41cbafce65930a22a6e8","a0865121f83b41cbafce65930a22a6e7","a0865121f83b41cbafce65930a22a6e6"]
product_id	String	是	产品标识。 获取方法，请参考**查询产品规格列表**。
kafka_manager_user	String	是	表示登录Kafka Manager的用户名。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。
kafka_manager_password	String	是	表示登录Kafka Manager的密码。 复杂度要求： ● 输入长度为8到32位的字符串。 ● 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（`~!@#$%^&*()-_=+|[{}]:'",<.>/?）
maintain_begin	String	否	维护时间窗开始时间，格式为HH:mm:ss。 ● 维护时间窗开始和结束时间必须为指定的时间段，可参考**查询维护时间窗时间段**获取。 ● 开始时间必须为22:00:00、02:00:00、06:00:00、10:00:00、14:00:00和18:00:00。 ● 该参数不能单独为空，若该值为空，则结束时间也为空。系统分配一个默认开始时间02:00:00。

参数	类型	是否必选	说明
maintain_end	String	否	维护时间窗结束时间，格式为HH:mm:ss。 ● 维护时间窗开始和结束时间必须为指定的时间段，可参考**查询维护时间窗时间段**获取。 ● 结束时间在开始时间基础上加四个小时，即当开始时间为22:00:00时，结束时间为02:00:00。 ● 该参数不能单独为空，若该值为空，则开始时间也为空，系统分配一个默认结束时间06:00:00。
enable_publicip	Boolean	否	实例是否开启公网访问功能。 ● true：开启 ● false：不开启
public_bandwidth	String	否	表示公网带宽，单位是Mbit/s。 取值范围： ● Kafka实例规格为100MB时，公网带宽取值范围3到900，且必须为实例节点个数的倍数。 ● Kafka实例规格为300MB时，公网带宽取值范围3到900，且必须为实例节点个数的倍数。 ● Kafka实例规格为600MB时，公网带宽取值范围4到1200，且必须为实例节点个数的倍数。 ● Kafka实例规格为1200MB时，公网带宽取值范围8到2400，且必须为实例节点个数的倍数。
publicip_id	String	否	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。
ssl_enable	Boolean	否	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。
retention_policy	String	否	磁盘的容量到达容量阈值后，对于消息的处理策略。取值如下： ● time_base：自动删除 ● produce_reject：生产受限

参数	类型	是否必选	说明
enable_auto_topic	Boolean	否	是否打开Kafka自动创建Topic功能。 ● true：开启 ● false：关闭 当您选择开启，表示生产或消费一个未创建的Topic时，会自动创建一个包含3个分区和3个副本的Topic。
storage_spec_code	String	是	存储IO规格。如何选择磁盘类型请参考**磁盘类型及性能介绍**。 取值范围： ● 参数specification为100MB时，取值dms.physical.storage.high或者dms.physical.storage.ultra ● 参数specification为300MB时，取值dms.physical.storage.high或者dms.physical.storage.ultra ● 参数specification为600MB时，取值dms.physical.storage.ultra ● 参数specification为1200MB时，取值dms.physical.storage.ultra
enterprise_project_id	String	否	企业项目ID。
tags	Array<Object>	否	标签列表。

表 7-3 tags

参数	参数类型	是否必选	描述
key	String	否	键。最大长度36个unicode字符。 key不能为空，不能为空字符串。 不能包含下列字符：非打印字符ASCII(0-31)，“=”，“*”，“<”，“>”，“\”，“,”，“|”，“/”。
value	String	否	值。每个值最大长度43个unicode字符。 value不能为空，可以空字符串。 不能包含下列字符：非打印字符ASCII(0-31)，“=”，“*”，“<”，“>”，“\”，“,”，“|”，“/”。

```
{
 "name": "kafka-test",
 "engine": "kafka",
 "engine_version": "2.3.0",
 "specification": "100MB",
 "storage_space": 600,
 "partition_num": 300,
 "vpc_id": "b50c1aa7-39e0-420e-936b-ee5d35288f9c",
 "security_group_id": "d8c81e0f-de6a-4110-8c96-81af3eacb3d1",
 "subnet_id": "0b6cfaea-bce7-48eb-b38d-267c24df5f79",
 "available_zones": [
  "38b0f7a602344246bcb0da47b5d548e7"
 ],
 "product_id": "00300-30308-0--0",
 "kafka_manager_user": "test",
 "kafka_manager_password": "Zxxxx",
 "enable_publicip": true,
 "publicip_id": "87864b85-7097-4c06-9d62-718d7359a503,72c12ba7-fade-4b06-
a680-01d335cf786d,11b535df-ed6d-4521-8d00-12bb60beb617",
 "storage_spec_code": "dms.physical.storage.high"
}
```

响应消息

响应参数

参数说明见**表7-4**。

表 7-4 参数说明

参数	类型	说明
instance_id	String	实例ID

响应示例

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

状态码

操作成功的状态码如**表7-5**所示，其他响应见**状态码**。

表 7-5 状态码

状态码	描述
200	创建实例成功。

7.1.1.2 查询指定实例

说明

当前页面API为历史版本API，未来可能停止维护。请使用**查询指定实例**。

功能介绍

查询指定实例的详细信息。

URI

GET /v1.0/{project_id}/instances/{instance_id}

参数说明见表7-6。

表 7-6 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID。

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}
```

响应消息

响应参数

参数说明见表7-7。

表 7-7 参数说明

参数	类型	说明
name	String	实例名称。
engine	String	消息引擎。
engine_version	String	消息引擎版本。
specification	String	实例规格。
storage_space	Integer	消息存储空间，单位：GB。
partition_num	String	Kafka实例的总分区数。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
connect_address	String	实例连接IP地址。

参数	类型	说明
port	Integer	实例连接端口。
status	String	实例的状态。详细状态说明见**实例状态说明**。
instance_id	String	实例ID。
resource_spec_code	String	资源规格标识。 • dms.instance.kafka.cluster.c3.mini：Kafka实例的基准带宽为100MByte/秒。 • dms.instance.kafka.cluster.c3.small.2：Kafka实例的基准带宽为300MByte/秒。 • dms.instance.kafka.cluster.c3.middle.2：Kafka实例的基准带宽为600MByte/秒。 • dms.instance.kafka.cluster.c3.high.2：Kafka实例的基准带宽为1200MByte/秒。
type	String	实例类型：集群，cluster
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	VPC ID。
vpc_name	String	VPC的名称。
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	子网网段。
available_zones	Array	实例节点所在的可用区，返回“可用区ID”。
user_id	String	用户id。
user_name	String	用户名。
access_user	String	实例的用户名。
order_id	String	订单ID。
maintain_begin	String	维护时间窗开始时间，格式为HH:mm:ss。

参数	类型	说明
maintain_end	String	维护时间窗结束时间，格式为HH:mm:ss。
enable_publicip	Boolean	实例是否开启公网访问功能。 ● true：开启 ● false：未开启
management_connect_address	String	Kafka实例的Kafka Manager连接地址。
ssl_enable	Boolean	是否开启安全认证。 ● true：开启 ● false：未开启
enterprise_project_id	String	企业项目ID。
is_logical_volume	Boolean	实例扩容时用于区分老实例与新实例。 ● true：新创建的实例，允许磁盘动态扩容不需要重启。 ● false：老实例。
extend_times	Integer	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。
enable_auto_topic	Boolean	是否打开Kafka自动创建Topic功能。 ● true：开启 ● false：关闭
total_storage_space	Integer	总共消息存储空间，单位：GB。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	IO规格。
service_type	String	服务类型。
storage_type	String	存储类型。
retention_policy	String	消息老化策略。
kafka_public_status	String	Kafka公网开启状态。
public_bandwidth	Integer	公网带宽。
public_connect_address	String	实例公网连接IP地址。当实例开启了公网访问，实例才包含该参数。

参数	类型	说明
kafka_manager_user	String	登录Kafka Manager的用户名。
enable_log_collection	Boolean	是否开启消息收集功能。
cross_vpc_info	String	跨VPC访问信息。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
rest_enable	Boolean	是否开启Kafka rest功能。
rest_connect_address	String	Kafka rest连接地址。
message_query_inst_enable	Boolean	是否开启消息查询功能。
vpc_client_plain	Boolean	是否开启VPC明文访问。
support_features	String	Kafka实例支持的特性功能。
trace_enable	Boolean	是否开启消息轨迹功能。
pod_connect_address	String	租户侧连接地址。
disk_encrypted	Boolean	是否开启磁盘加密。 • 是：开启 • 否：不开启
kafka_private_connect_address	String	Kafka实例私有连接地址。
ces_version	String	云监控版本。
tags	Array<Object>	标签列表。

表 7-8 tags

参数	参数类型	描述
key	String	标签的键。
value	String	标签的值。

响应示例

```
{
    "name": "kafka-l00230526",
    "engine": "kafka",
    "port": 9092,
    "status": "RUNNING",
    "type": "cluster",
    "specification": "100MB",
    "engine_version": "XXX",
    "connect_address": "192.168.1.116,192.168.1.152,192.168.1.78",
    "connect_dn": "",
    "instance_id": "ef84dd5f-3ece-4336-8c99-987defd62e3a",
    "resource_spec_code": "dms.instance.kafka.cluster.c3.mini",
    "charging_mode": 1,
    "vpc_id": "2477879f-aebf-496f-a08a-67812885ce9b",
    "vpc_name": "vpc-y00502467",
    "created_at": "1568797295209",
    "product_id": "00300-30308-0--0",
    "security_group_id": "008a08e2-10cc-4d9b-90ab-3f3b8f6c3333",
    "security_group_name": "z00417080-cce-node-na7j",
    "subnet_id": "5ca08fb7-7522-4d95-9fa5-ff6b3592a29d",
    "subnet_name": "subnet-cyd-6102",
    "subnet_cidr": "192.168.1.0/24",
    "available_zones": [
        "ae04cf9d61544df3806a3feeb401b204"
    ],
    "user_id": "2b4af4428ec840dfa1f0f1a32e965567",
    "user_name": "laiyh",
    "kafka_manager_user": "root",
    "maintain_begin": "22:00:00",
    "maintain_end": "02:00:00",
    "storage_space": 492,
    "total_storage_space": 600,
    "used_storage_space": 25,
    "partition_num": "300",
    "ssl_enable": false,
    "management_connect_address": "https://192.168.1.116:9999",
    "storage_resource_id": "81982562-ce8b-490a-95fa-2b225c292271",
    "storage_spec_code": "dms.physical.storage.ultra",
    "service_type": "advanced",
    "storage_type": "hec",
    "enterprise_project_id": "0",
    "is_logical_volume": true,
    "extend_times": 0,
    "retention_policy": "produce_reject",
    "ipv6_enable": false,
    "ipv6_connect_addresses": [],
    "connector_enable": false,
    "connector_id": "",
    "rest_enable": false,
    "rest_connect_address": "",
    "message_query_inst_enable": true,
    "vpc_client_plain": false,
    "support_features":
"feature.physerver.kafka.topic.accesspolicy,message_trace_enable,features.pod.token.access,feature.physerver.
kafka.pulbic.dynamic,feature.physerver.kafka.user.manager",
    "trace_enable": false,
    "agent_enable": false,
    "pod_connect_address": "100.113.16.105:9100,100.113.5.197:9100,100.113.15.231:9100",
    "disk_encrypted": false,
    "enable_auto_topic": true
}
```


状态码

操作成功的状态码如**表7-9**所示，其他响应见**状态码**。

表 7-9 状态码

状态码	描述
200	查询指定实例成功。

7.1.1.3 修改实例信息

说明

当前页面API为历史版本API，未来可能停止维护。请使用**修改实例信息**。

功能介绍

修改实例的信息，包括实例名称、实例描述、实例维护时间窗、安全组等信息。

URI

PUT /v1.0/{project_id}/instances/{instance_id}

表 7-10 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID。

请求消息

请求参数

参数说明见**表7-11**。

表 7-11 参数说明

参数	类型	必选	说明
name	String	否	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	String	否	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\"。

参数	类型	必选	说明
maintain_begin	String	否	维护时间窗开始时间，格式为HH:mm:ss。 ● 维护时间窗开始和结束时间必须为指定的时间段，可参考**查询维护时间窗时间段**。 ● 开始时间必须为22:00:00、02:00:00、06:00:00、10:00:00、14:00:00和18:00:00。 ● 该参数不能单独为空，若该值为空，则结束时间也为空。系统分配一个默认开始时间02:00:00。
maintain_end	String	否	维护时间窗结束时间，格式为HH:mm:ss。 ● 维护时间窗开始和结束时间必须为指定的时间段，可参考**查询维护时间窗时间段**。 ● 结束时间在开始时间基础上加四个小时，即当开始时间为22:00:00时，结束时间为02:00:00。 ● 该参数不能单独为空，若该值为空，则开始时间也为空。系统分配一个默认结束时间06:00:00。
security_group_id	String	否	安全组ID。
retention_policy	String	否	容量阈值策略。 支持两种策略模式： ● produce_reject: 生产受限 ● time_base: 自动删除
enterprise_project_id	String	否	企业项目。

请求示例

示例1 ：

```
PUT https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}
{
   "name": "dms002",
   "description": "instance description"
}
```

示例2

```
PUT https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}
{
    "name": "dms002",
    "description": "instance description",
    "maintain_begin":"02:00:00",
    "maintain_end":"06:00:00"
}
```

响应消息

响应参数

无。

响应样例

无。

状态码

操作成功的状态码如表7-12所示，其他响应见状态码。

表 7-12 状态码

状态码	描述
204	修改实例成功。

7.1.1.4 删除指定实例

说明

当前页面API为历史版本API，未来可能停止维护。请使用删除指定的实例。

功能介绍

删除指定的实例，释放该实例的所有资源。

URI

DELETE /v1.0/{project_id}/instances/{instance_id}

参数说明见表7-13。

表 7-13 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID。

请求消息

请求参数

无。

请求示例

```
DELETE https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}
```

响应消息

响应参数

无。

响应示例

无。

状态码

操作成功的状态码如表7-14所示，其他响应见状态码。

表 7-14 状态码

状态码	描述
204	删除实例成功。

7.1.1.5 批量重启、删除实例

说明

当前页面API为历史版本API，未来可能停止维护。请使用批量重启或删除实例。

功能介绍

批量重启或删除实例。

在实例重启过程中，客户端的生产与消费消息等请求会被拒绝。

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。

URI

POST /v1.0/{project_id}/instances/action

参数说明见表7-15。

表 7-15 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。

请求消息

请求参数

参数说明见表7-16。

表 7-16 参数说明

参数	类型	必选	说明
action	String	是	对实例的操作：restart、delete
instances	Array	否	实例的ID列表。
allFailure	String	否	参数值为kafka，表示删除租户所有创建失败的Kafka实例。

请求示例

批量重启实例

```
POST https://{dms_endpoint}/v1.0/{project_id}/instances/action
{
    "action" : "restart",
    "instances" : ["54602a9d-5e22-4239-9123-77e350df4a34", "7166cdea-dbad-4d79-9610-7163e6f8b640"]
}
```

批量删除实例

```
POST https://{dms_endpoint}/v1.0/{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实例

```
POST https://{dms_endpoint}/v1.0/{project_id}/instances/action
{
    "action" : "delete",
    "allFailure" : "kafka"
}
```

响应消息

响应参数

当参数action为delete，allFailure值为kafka时，响应返回为空表示删除成功。参数说明见表7-17。

表 7-17 参数说明

参数	类型	说明
results	Array	修改实例的结果。

表 7-18 results 参数说明

参数	类型	说明
instance	String	实例ID。

参数	类型	说明
result	String	操作结果：success、failed。

响应示例

```
{
   "results": [
      {
         "result": "success",
         "instance": "afc90a2a-a02c-4cba-94d5-58dfa9ad1e0d"
      },
      {
         "result": "success",
         "instance": "67fc5f8d-3986-4f02-bb75-4075a23112de"
      }
   ]
}
```

状态码

操作成功的状态码如表7-19所示，其他响应见状态码。

表 7-19 状态码

状态码	描述
200	重启或者删除实例成功。
204	删除创建失败实例成功。

7.1.1.6 查询所有实例列表

说明

当前页面API为历史版本API，未来可能停止维护。请使用查询所有实例列表。

功能介绍

查询租户的实例列表，支持按照条件查询。

URI

GET /v1.0/{project_id}/instances?engine={engine}&name={name}&status={status}&id={id}&includeFailure={includeFailure}&exactMatchName={exactMatchName}&enterprise_project_id={enterprise_project_id}

参数说明见表7-20。

表 7-20 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
engine	String	否	消息引擎：kafka。
name	String	否	实例名称。
id	String	否	实例ID。
status	String	否	实例状态。详细状态说明见**实例状态说明**。
includeFailure	String	否	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。
exactMatchName	String	否	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	String	否	企业项目ID。

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/{project_id}/instances?
start=1&limit=10&name=&status=&id=&includeFailure=true&exactMatchName=false
```

响应消息

响应参数

参数说明见**表7-21**。

表 7-21 参数说明

参数	类型	说明
instances	Array	实例的详情数组。
instance_num	Integer	实例个数。

表 7-22 instance 参数说明

参数	类型	说明
name	String	实例名称。
engine	String	引擎。
engine_version	String	版本。
specification	String	实例规格。
storage_space	Integer	消息存储空间，单位：GB。
partition_num	String	Kafka实例的最大topic数。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
connect_address	String	实例连接IP地址。
port	Integer	实例连接端口。
status	String	实例的状态。详细状态说明见**实例状态说明**。
instance_id	String	实例ID。
resource_spec_code	String	资源规格标识。 ● dms.instance.kafka.cluster.c3.mini：Kafka实例的基准带宽为100MB。 ● dms.instance.kafka.cluster.c3.small.2：Kafka实例的基准带宽为300MB。 ● dms.instance.kafka.cluster.c3.middle.2：Kafka实例的基准带宽为600MB。 ● dms.instance.kafka.cluster.c3.high.2：Kafka实例的基准带宽为1200MB。
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	VPC ID。
vpc_name	String	VPC的名称。
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。
user_id	String	用户id。
user_name	String	用户名。
access_user	String	实例的用户名。
order_id	String	订单ID。

参数	类型	说明
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_publicip	Boolean	实例是否开启公网访问功能。 ● true：开启 ● false：未开启
management_connect_address	String	Kafka实例的Kafka Manager连接地址。
ssl_enable	Boolean	是否开启安全认证。 ● true：开启 ● false：未开启
enterprise_project_id	String	企业项目ID。
is_logical_volume	Boolean	实例扩容时用于区分老实例与新实例。 ● true：新创建的实例，允许磁盘动态扩容不需要重启。 ● false：老实例。
extend_times	Integer	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。
enable_auto_topic	Boolean	是否打开kafka自动创建topic功能。 ● true：开启 ● false：关闭
type	String	实例类型：集群，cluster。
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。
available_zones	Array	实例节点所在的可用区，返回“可用区ID”。
total_storage_space	Integer	总共消息存储空间，单位：GB。
public_connect_address	String	实例公网连接IP地址。当实例开启了公网访问，实例才包含该参数。
storage_resource_id	String	存储资源ID。

参数	类型	说明
storage_spec_code	String	IO规格。
service_type	String	服务类型。
storage_type	String	存储类型。
retention_policy	String	消息老化策略。
kafka_public_status	String	Kafka公网开启状态。
public_bandwidth	Integer	公网带宽。
kafka_manager_user	String	登录Kafka Manager的用户名。
enable_log_collection	Boolean	是否开启消息收集功能。
cross_vpc_info	String	跨VPC访问信息。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
rest_enable	Boolean	是否开启Kafka rest功能。
rest_connect_address	String	Kafka rest地址。
message_query_inst_enable	Boolean	是否开启消息查询功能。
vpc_client_plain	Boolean	是否开启VPC明文访问。
support_features	String	Kafka实例支持的特性功能。
trace_enable	Boolean	是否开启消息轨迹功能。
pod_connect_address	String	租户侧连接地址。
disk_encrypted	Boolean	是否开启磁盘加密。
kafka_private_connect_address	String	Kafka实例私有连接地址。
ces_version	String	云监控版本。
tags	Array<Object>	标签列表。

表 7-23 tags

参数	参数类型	描述
key	String	标签的键。
value	String	标签的值。

响应示例


```
{
   "instances": [
      {
         "name": "kafka-l00230526",
         "engine": "kafka",
         "port": 9092,
         "status": "RUNNING",
         "type": "cluster",
         "specification": "100MB",
         "engine_version": "XXX",
         "connect_address": "192.168.1.116,192.168.1.152,192.168.1.78",
         "instance_id": "ef84dd5f-3ece-4336-8c99-987defd62e3a",
         "resource_spec_code": "dms.instance.kafka.cluster.c3.mini",
         "charging_mode": 1,
         "vpc_id": "2477879f-aebf-496f-a08a-67812885ce9b",
         "vpc_name": "vpc-y00502467",
         "created_at": "1568797295209",
         "product_id": "00300-30308-0--0",
         "security_group_id": "008a08e2-10cc-4d9b-90ab-3f3b8f6c3333",
         "security_group_name": "z00417080-cce-node-na7j",
         "subnet_id": "5ca08fb7-7522-4d95-9fa5-ff6b3592a29d",
         "available_zones": [
            "ae04cf9d61544df3806a3feeb401b204"
         ],
         "user_id": "2b4af4428ec840dfa1f0f1a32e965567",
         "user_name": "laiyh",
         "kafka_manager_user": "root",
         "maintain_begin": "22:00",
         "maintain_end": "02:00",
         "storage_space": 492,
         "total_storage_space": 600,
         "used_storage_space": 25,
         "partition_num": "300",
         "ssl_enable": false,
         "management_connect_address": "https://192.168.1.116:9999",
         "storage_resource_id": "81982562-ce8b-490a-95fa-2b225c292271",
         "storage_spec_code": "dms.physical.storage.ultra",
         "service_type": "advanced",
         "storage_type": "hec",
         "enterprise_project_id": "0",
         "is_logical_volume": true,
         "extend_times": 0,
         "retention_policy": "produce_reject",
         "ipv6_enable": false,
         "ipv6_connect_addresses": [],
         "rest_enable": false,
         "rest_connect_address": "",
         "message_query_inst_enable": true,
         "vpc_client_plain": false,
         "support_features":
"feature.physerver.kafka.topic.accesspolicy,message_trace_enable,features.pod.token.access,feature.physerver.
kafka.pulbic.dynamic,feature.physerver.kafka.user.manager",
         "trace_enable": false,
```

```
            "agent_enable": false,
            "pod_connect_address": "100.113.16.105:9100,100.113.5.197:9100,100.113.15.231:9100",
            "disk_encrypted": false,
            "enable_auto_topic": true
        }
    ],
    "instance_num": 1
}
```

状态码

操作成功的状态码如表7-24所示，其他响应见状态码。

表 7-24 状态码

状态码	描述
200	查询所有实例列表成功。

7.1.1.7 Kafka 实例创建 Topic

说明

当前页面API为历史版本API，未来可能停止维护。请使用Kafka实例创建Topic。

功能介绍

该接口用于向Kafka实例创建Topic。

URI

POST /v1.0/{project_id}/instances/{instance_id}/topics

参数说明见表7-25。

表 7-25 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID

请求消息

请求参数

参数说明见表7-26。

表 7-26 参数说明

参数	类型	是否必选	说明
id	String	是	topic名称，长度为4-64，以字母开头且只支持大小写字母、中横线、下划线以及数字。
partition	Integer	否	topic分区数，设置消费的并发数。 默认值为3，取值范围为1~100。
replication	Integer	否	副本数，配置数据的可靠性 默认值为3，取值范围为1~3。
sync_replication	Boolean	否	是否开启同步复制，开启后，客户端生产消息时相应的也要设置acks=-1，否则不生效。 默认关闭。
retention_time	Integer	否	消息老化时间。默认值为72。 取值范围1~720，单位小时。
sync_message_flush	Boolean	否	是否使用同步落盘。默认值为false。同步落盘会导致性能降低。

请求示例

```
POST https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}/topics
{
  "id" : "haha",
  "partition" : 3,
  "replication" : 3,
  "sync_replication " : true,
  "retention_time" : 10,
  "sync_message_flush" : true
}
```

响应消息

响应参数

参数说明见**表7-27**。

表 7-27 参数说明

参数	类型	说明
id	String	topic名称

响应示例

```
{
"id": "haha"
}
```

状态码

操作成功的状态码如表7-28所示，其他响应见状态码。

表 7-28 状态码

状态码	描述
200	创建成功。

7.1.1.8 Kafka 实例查询 Topic

说明

当前页面API为历史版本API，未来可能停止维护。请使用Kafka实例查询Topic。

功能介绍

该接口用于查询指定Kafka实例的Topic详情。

URI

GET /v1.0/{project_id}/instances/{instance_id}/topics

参数说明见表7-29。

表 7-29 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}/topics
```

响应消息

响应参数

参数说明见表7-30。

表 7-30 参数说明

参数	类型	说明
total	Integer	topic总数。
size	Integer	分页查询的大小。
remain_partitions	Integer	剩余分区数。
max_partitions	Integer	分区总数。
topics	Array	Topic列表。

表 7-31 参数说明

参数	类型	说明
policiesOnly	Boolean	是否为默认策略。
id	String	Topic名称
replication	Integer	副本数，配置数据的可靠性
partition	Integer	Topic分区数，设置消费的并发数。
retention_time	Integer	消息老化时间。
sync_replication	Boolean	是否开启同步复制，开启后，客户端生产消息时相应的也要设置acks=-1，否则不生效。 默认关闭。
sync_message_flush	Boolean	是否使用同步落盘。同步落盘会导致性能降低。
external_configs	Object	扩展配置。
topic_type	Integer	Topic类型。

响应示例

```
{
 "count": 1,
 "topics": [
  {
   "id": "topic-test",
   "replication": 3,
   "partition": 4,
   "retention_time": 72,
   "sync_replication": "false",
   "sync_message_flush": "false"
  }
 ]
}
```

状态码

操作成功的状态码如表7-32所示，其他响应见状态码。

表 7-32 状态码

状态码	描述
200	查询成功。

7.1.1.9 Kafka 实例批量删除 Topic

说明

当前页面API为历史版本API，未来可能停止维护。请使用Kafka实例批量删除Topic。

功能介绍

该接口用于向Kafka实例批量删除Topic。

URI

POST /v1.0/{project_id}/instances/{instance_id}/topics/delete

参数说明见表7-33。

表 7-33 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
instance_id	String	是	实例ID

请求消息

请求参数

参数说明见表7-34。

表 7-34 参数说明

参数	类型	是否必选	说明
topics	Array	是	待删除的topic列表

请求示例

```
POST https://{dms_endpoint}/v1.0/{project_id}/instances/{instance_id}/topics/delete
{
  "topics" : ["hah", "aabb"]
 }
```

响应消息

响应参数

参数说明见**表7-35**。

表 7-35 参数说明

参数	类型	说明
topics	Array	Topic列表

表 7-36 topics 参数说明

参数	类型	说明
id	String	Topic名称
success	Boolean	是否删除成功。

响应示例

```
{
  "topics" : [{
      "id" : "haha",
      "success" : true
    }, {
      "id" : "aabb",
      "success" : true
    }
  ]
}
```

状态码

操作成功的状态码如**表7-37**所示，其他响应见**状态码**。

表 7-37 状态码

状态码	描述
200	删除成功。

7.1.2 其他接口

7.1.2.1 查询可用区信息

说明

当前页面API为历史版本API，未来可能停止维护。请使用**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。

URI

GET /v1.0/availableZones

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/availableZones
```

响应消息

响应参数

参数说明见**表7-38**、**表7-39**。

表 7-38 参数说明

参数	类型	说明
region_id	String	区域ID。
available_zones	Array	可用区数组，具体请参考**表7-39**。

表 7-39 available_zones 参数说明

参数	类型	说明
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。

参数	类型	说明
resource_availability	String	分区上是否还有可用资源。 ● true：还有资源。 ● false：资源已售罄。

响应示例

```
{
   regionId: "XXXX",
   available_zones:[
     {
         "id":"1d7b939b382c4c3bb3481a8ca10da768",
         "name":"az10.dc1",
         "code":"az10.dc1",
         "port":"8002",
         "resource_availability": "true"
      },
      {
         "id":"1d7b939b382c4c3bb3481a8ca10da769",
         "name":"az10.dc2",
         "code":"az10.dc2",
         "port":"8002",
         "resource_availability": "true"
      }
   ]
}
```

状态码

操作成功的状态码如**表7-40**所示，其他响应见**状态码**。

表 7-40 状态码

状态码	描述
200	查询成功。

7.1.2.2 查询产品规格列表

说明

当前页面API为历史版本API，未来可能停止维护。请使用**查询产品规格列表**。

功能介绍

在创建kafka实例时，需要配置订购的产品ID（即product_id），可通过该接口查询产品规格。

例如，要订购按需计费、基准带宽为100MB的kafka实例，可从接口响应消息中，查找Hourly的消息体，然后找到bandwidth为100MB的记录对应的product_id，该product_id的值即是创建上述kafka实例时需要配置的产品ID。

同时，unavailable_zones字段表示资源不足的可用区列表，如果为空，则表示所有可用区都有资源，如果不为空，则表示字段值的可用区没有资源。所以必须确保您购买的资源所在的可用区有资源，不在该字段列表内。

URI

GET /v1.0/products?engine={engine}

参数说明见表7-41。

表 7-41 参数说明

参数	类型	必选	说明
engine	String	是	消息引擎的类型。

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/products?engine={engine}
```

响应消息

响应参数

Hourly或者Monthly的参数说明见表7-43。

表 7-42 参数说明

参数	类型	说明
Hourly	Array	表示按需付费的产品列表。
Monthly	Array	表示包年包月的产品列表。当前暂不支持通过API创建包年包月的kafka实例。

表 7-43 参数说明

参数	类型	说明
name	String	消息引擎的名称，该字段显示为kafka。
version	String	消息引擎的版本。
values	Array	产品规格列表。具体参数，请参见表7-44。

表 7-44 values 参数说明

参数	类型	说明
detail	Array	规格详情。具体参数，请参见表7-45。
name	String	实例类型。
unavailable_zones	Array	资源售罄的可用区。
available_zones	Array	有可用资源的可用区。

表 7-45 Kafka 实例的 detail 参数说明

参数	类型	说明
tps	String	单位时间内的消息量最大值。
storage	String	消息存储空间。
partition_num	String	Kafka实例的最大Topic数。
product_id	String	产品ID。
spec_code	String	规格ID。
io	Array	IO信息。具体参数，请参见表7-46。
bandwidth	String	Kafka实例的基准带宽。
available_zones	Array	实例资源未售罄的可用区。
ecs_flavor_id	String	该产品规格对应的虚拟机规格。
arch_type	String	实例规格架构类型。当前仅支持X86。

表 7-46 io 参数说明

参数	类型	说明
io_type	String	IO类型。
storage_spec_code	String	IO规格。
available_zones	Array	IO未售罄的可用区。
unavailable_zones	Array of strings	IO已售罄的不可用区列表。
volume_type	String	磁盘类型。

响应示例

```
{
   "Hourly": [{
      "name": "kafka",
      "version": "XXX",
      "values": [{
         "detail": [{
            "tps": "50000",
            "storage": "600",
            "partition_num": "300",
            "product_id": "00300-30308-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.mini",
            "io": [{
               "io_type": "high",
               "storage_spec_code": "dms.physical.storage.high",
               "volume_type": "SAS"
            },
            {
               "io_type": "ultra",
               "storage_spec_code": "dms.physical.storage.ultra",
               "volume_type": "SSD"
            }],
            "bandwidth": "100MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.large.2"
         },
         {
            "tps": "100000",
            "storage": "1200",
            "partition_num": "900",
            "product_id": "00300-30310-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.small.2",
            "io": [{
               "io_type": "high",
               "storage_spec_code": "dms.physical.storage.high",
               "volume_type": "SAS"
            },
            {
               "io_type": "ultra",
               "storage_spec_code": "dms.physical.storage.ultra",
               "volume_type": "SSD"
            }],
            "bandwidth": "300MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.xlarge.2"
         },
         {
            "tps": "200000",
            "storage": "2400",
            "partition_num": "1800",
            "product_id": "00300-30312-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.middle.2",
            "io": [{
               "io_type": "ultra",
               "storage_spec_code": "dms.physical.storage.ultra",
               "volume_type": "SSD"
            }],
            "bandwidth": "600MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.2xlarge.2"
         },
         {
            "tps": "300000",
            "storage": "4800",
            "partition_num": "1800",
            "product_id": "00300-30314-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.high.2",
            "io": [{
               "io_type": "ultra",
               "storage_spec_code": "dms.physical.storage.ultra",
```

```
                "volume_type": "SSD"
            }],
            "bandwidth": "1200MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3ne.2xlarge.2"
        }],
        "name": "cluster",
        "unavailable_zones": []
    }]
}],
"Monthly": [{
    "name": "kafka",
    "version": "XXX",
    "values": [{
        "detail": [{
            "tps": "50000",
            "storage": "600",
            "partition_num": "300",
            "product_id": "00300-30309-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.mini",
            "io": [{
                "io_type": "high",
                "storage_spec_code": "dms.physical.storage.high",
                "volume_type": "SAS"
            },
            {
                "io_type": "ultra",
                "storage_spec_code": "dms.physical.storage.ultra",
                "volume_type": "SSD"
            }],
            "bandwidth": "100MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.large.2"
        },
        {
            "tps": "100000",
            "storage": "1200",
            "partition_num": "900",
            "product_id": "00300-30311-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.small.2",
            "io": [{
                "io_type": "high",
                "storage_spec_code": "dms.physical.storage.high",
                "volume_type": "SAS"
            },
            {
                "io_type": "ultra",
                "storage_spec_code": "dms.physical.storage.ultra",
                "volume_type": "SSD"
            }],
            "bandwidth": "300MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.xlarge.2"
        },
        {
            "tps": "200000",
            "storage": "2400",
            "partition_num": "1800",
            "product_id": "00300-30313-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.middle.2",
            "io": [{
                "io_type": "ultra",
                "storage_spec_code": "dms.physical.storage.ultra",
                "volume_type": "SSD"
            }],
            "bandwidth": "600MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3.2xlarge.2"
        },
```



```
        {
            "tps": "300000",
            "storage": "4800",
            "partition_num": "1800",
            "product_id": "00300-30315-0--0",
            "spec_code": "dms.instance.kafka.cluster.c3.high.2",
            "io": [{
                "io_type": "ultra",
                "storage_spec_code": "dms.physical.storage.ultra",
                "volume_type": "SSD"
            }],
            "bandwidth": "1200MB",
            "unavailable_zones": [],
            "ecs_flavor_id": "c3ne.2xlarge.2"
        }],
        "name": "cluster",
        "unavailable_zones": []
    }]
  }]
}
```


状态码

操作成功的状态码如表7-47所示，其他响应见状态码。

表 7-47 状态码

状态码	描述
200	查询规格列表成功。

7.1.2.3 查询维护时间窗时间段

说明

当前页面API为历史版本API，未来可能停止维护。请使用查询维护时间窗时间段。

功能介绍

查询维护时间窗开始时间和结束时间。

URI

GET /v1.0/instances/maintain-windows

请求消息

请求参数

无。

请求示例

```
GET https://{dms_endpoint}/v1.0/instances/maintain-windows
```

响应消息

响应参数

参数说明见表**表7-48**、**表7-49**

表 7-48 响应参数说明

参数	类型	说明
maintain_windows	Array	支持的维护时间窗列表

表 7-49 maintain_windows 参数说明

参数	类型	说明
seq	Integer	序号。
begin	String	维护时间窗开始时间。
end	String	维护时间窗结束时间。
default	Boolean	是否为默认时间段。

响应示例

```
{
   "maintain_windows": [{
      "default": false,
      "seq": 1,
      "begin": "22:00:00",
      "end": "02:00:00"
   },
   {
      "default": true,
      "seq": 2,
      "begin": "02:00:00",
      "end": "06:00:00"
   },
   {
      "default": false,
      "seq": 3,
      "begin": "06:00:00",
      "end": "10:00:00"
   },
   {
      "default": false,
      "seq": 4,
      "begin": "10:00:00",
      "end": "14:00:00"
   },
   {
      "default": false,
      "seq": 5,
      "begin": "14:00:00",
      "end": "18:00:00"
   },
   {
      "default": false,
      "seq": 6,
      "begin": "18:00:00",
      "end": "22:00:00"
```

```
  }]
}
```

状态码

操作成功的状态码如表7-50所示，其他响应见状态码。

表 7-50 状态码

状态码	描述
200	查询维护时间窗时间段成功。

7.2 API V2

7.2.1 生命周期管理

7.2.1.1 查询指定实例

功能介绍

查询指定实例的详细信息。

说明

当前页面API为历史版本API，未来可能停止维护。请使用查询指定实例。

调用方法

请参见如何调用API。

URI

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

表 7-51 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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 7-52 响应 Body 参数

参数	参数类型	描述
name	String	实例名称。
engine	String	引擎。
engine_version	String	版本。
description	String	实例描述。
specification	String	实例规格。
storage_space	Integer	消息存储空间，单位：GB。
partition_num	String	Kafka实例的分区数量。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
dns_enable	Boolean	实例是否开启域名访问功能。 ● true：开启 ● false：未开启
connect_address	String	实例连接IP地址。
port	Integer	实例连接端口。
status	String	实例的状态。详细状态说明请参考**实例状态说明**。
instance_id	String	实例ID。
resource_spec_code	String	资源规格标识。 ● dms.instance.kafka.cluster.c3.mini：Kafka实例的基准带宽为100MByte/秒。 ● dms.instance.kafka.cluster.c3.small.2：Kafka实例的基准带宽为300MByte/秒。 ● dms.instance.kafka.cluster.c3.middle.2：Kafka实例的基准带宽为600MByte/秒。 ● dms.instance.kafka.cluster.c3.high.2：Kafka实例的基准带宽为1200MByte/秒。
charging_mode	Integer	付费模式，1表示按需计费，0表示包年/包月计费。

参数	参数类型	描述
vpc_id	String	VPC ID。
vpc_name	String	VPC的名称。
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。
subnet_name	String	子网名称。
subnet_cidr	String	子网网段。
user_id	String	用户ID。
user_name	String	用户名。
access_user	String	实例访问用户名。
order_id	String	订单ID，只有在包周期计费时才会有order_id值，其他计费方式order_id值为空。
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_publicip	Boolean	实例是否开启公网访问功能。 ● true：开启 ● false：未开启
management_connect_address	String	Kafka实例的Kafka Manager连接地址。
ssl_enable	Boolean	是否开启安全认证。 ● true：开启 ● false：未开启
broker_ssl_enable	Boolean	是否开启broker间副本加密传输。 ● true：开启 ● false：未开启
kafka_security_protocol	String	开启SASL后使用的安全协议。 ● SASL_SSL: 采用SSL证书进行加密传输，支持账号密码认证，安全性更高。 ● SASL_PLAINTEXT: 明文传输，支持账号密码认证，性能更好，建议使用SCRAM-SHA-512机制。

参数	参数类型	描述
sasl_enabled_mechanisms	Array of strings	开启SASL后使用的认证机制。 ● PLAIN: 简单的用户名密码校验。 ● SCRAM-SHA-512: 用户凭证校验，安全性比PLAIN机制更高。
ssl_two_way_enable	Boolean	是否开启双向认证。
cert_replaced	Boolean	是否能够证书替换。
public_management_connect_address	String	公网访问Kafka Manager连接地址。
enterprise_project_id	String	企业项目ID。
is_logical_volume	Boolean	实例扩容时用于区分老实例与新实例。 ● true：新创建的实例，允许磁盘动态扩容不需要重启。 ● false：老实例
extend_times	Integer	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。
enable_auto_topic	Boolean	是否打开kafka自动创建topic功能。 ● true：开启 ● false：关闭
type	String	实例类型：集群，cluster。
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。
available_zones	Array of strings	实例节点所在的可用区，返回“可用区ID”。
available_zone_names	Array of strings	实例节点所在的可用区名称，返回“可用区名称”。
total_storage_space	Integer	总共消息存储空间，单位：GB。
public_connect_address	String	实例公网连接IP地址。当实例开启了公网访问，实例才包含该参数。

参数	参数类型	描述
public_connect_domain_name	String	实例公网连接域名。当实例开启了公网访问，实例才包含该参数。
storage_resource_id	String	存储资源ID。
storage_spec_code	String	IO规格。
service_type	String	服务类型。
storage_type	String	存储类型。
retention_policy	String	消息老化策略。
kafka_public_status	String	Kafka公网开启状态。
public_bandwidth	Integer	kafka公网访问带宽。
enable_log_collection	Boolean	是否开启消息收集功能。
new_auth_cert	Boolean	是否开启新证书。
cross_vpc_info	String	跨VPC访问信息。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
connector_enable	Boolean	是否开启转储。新规格产品暂不支持开启转储。
connector_node_num	Integer	connector节点数量。
connector_id	String	转储任务ID。
rest_enable	Boolean	是否开启Kafka rest功能。
rest_connect_address	String	Kafka rest连接地址。
public_boundwidth	Integer	kafka公网访问带宽。待删除版本。
message_query_inst_enable	Boolean	是否开启消息查询功能。
vpc_client_plain	Boolean	是否开启VPC明文访问。
support_features	String	Kafka实例支持的特性功能。
trace_enable	Boolean	是否开启消息轨迹功能。
agent_enable	Boolean	是否开启代理。

参数	参数类型	描述
pod_connect_address	String	租户侧连接地址。
disk_encrypted	Boolean	是否开启磁盘加密。
disk_encrypted_key	String	磁盘加密key，未开启磁盘加密时为空。
kafka_private_connect_address	String	Kafka实例内网连接地址。
kafka_private_connect_domain_name	String	Kafka实例内网连接域名。
ces_version	String	云监控版本。
public_access_enabled	String	区分实例什么时候开启的公网访问 取值范围： • true：已开启公网访问 • actived：已开启公网访问 • closed：已关闭公网访问 • false：已关闭公网访问
node_num	Integer	节点数。
port_protocols	**PortProtocolsEntity** object	实例支持的连接方式及其连接地址。
enable_acl	Boolean	是否开启访问控制。
new_spec_billing_enable	Boolean	是否启用新规格计费。
broker_num	Integer	节点数量。
tags	Array of **TagEntity** objects	标签列表。
dr_enable	Boolean	是否为容灾实例。

表 7-53 PortProtocolsEntity

参数	参数类型	描述
private_plain_enable	Boolean	实例是否支持内网PLAINTEXT访问接入方式。 ● true：实例支持内网PLAINTEXT访问方式接入方式。 ● false：实例不支持内网PLAINTEXT访问接入方式。
private_plain_address	String	kafka内网PLAINTEXT接入方式连接地址。
private_plain_domain_name	String	内网明文连接域名
private_sasl_ssl_enable	Boolean	实例是否支持内网SASL_SSL访问接入方式。 ● true：实例支持内网SASL_SSL访问方式接入方式。 ● false：实例不支持内网SASL_SSL访问接入方式。
private_sasl_ssl_address	String	kafka内网SASL_SSL接入方式连接地址。
private_sasl_ssl_domain_name	String	内网SASL_SSL连接域名
private_sasl_plaintext_enable	Boolean	实例是否支持内网SASL_PLAINTEXT访问接入方式。 ● true，实例支持内网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持内网SASL_PLAINTEXT访问接入方式。
private_sasl_plaintext_address	String	kafka内网SASL_PLAINTEXT接入方式连接地址。
private_sasl_plaintext_domain_name	String	内网SASL_PLAINTEXT连接域名
public_plain_enable	Boolean	实例是否支持公网PLAINTEXT访问接入方式。 ● true，实例支持公网PLAINTEXT访问方式接入方式。 ● false，实例不支持公网PLAINTEXT访问接入方式。
public_plain_address	String	kafka公网PLAINTEXT接入方式连接地址。

参数	参数类型	描述
public_plain_domain_name	String	公网明文连接域名
public_sasl_ssl_enable	Boolean	实例是否支持公网SASL_SSL访问接入方式。 ● true，实例支持内网SASL_SSL访问方式接入方式。 ● false，实例不支持公网SASL_SSL访问接入方式。
public_sasl_ssl_address	String	kafka公网SASL_SSL接入方式连接地址。
public_sasl_ssl_domain_name	String	公网SASL_SSL连接域名
public_sasl_plaintext_enable	Boolean	实例是否支持公网SASL_PLAINTEXT访问接入方式。 ● true，实例支持公网SASL_PLAINTEXT访问方式接入方式。 ● false，实例不支持公网SASL_PLAINTEXT访问接入方式。
public_sasl_plaintext_address	String	kafka公网SASL_PLAINTEXT接入方式连接地址。
public_sasl_plaintext_domain_name	String	公网SASL_PLAINTEXT连接域名

表 7-54 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_ . : = + - @ ● 不能以_sys_开头。 ● 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_ . : = + - @

请求示例

查询指定实例。

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

响应示例

状态码： 200

查询指定实例成功。


```
{
  "name" : "kafka-2085975099",
  "engine" : "kafka",
  "port" : 9092,
  "status" : "RUNNING",
  "type" : "cluster",
  "specification" : "100MB",
  "engine_version" : "1.1.0",
  "connect_address" : "192.168.0.100,192.168.0.61,192.168.0.72",
  "instance_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "resource_spec_code" : "dms.instance.kafka.cluster.c3.mini",
  "charging_mode" : 1,
  "vpc_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "vpc_name" : "dms-test",
  "created_at" : "1585618587087",
  "product_id" : "00300-30308-0--0",
  "security_group_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "security_group_name" : "Sys-default",
  "subnet_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "available_zones" : [ "38b0f7a602344246bcb0da47b5d548e7" ],
  "available_zone_names" : [ "AZ1" ],
  "user_id" : "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
  "user_name" : "paas_dms",
  "access_user" : "root",
  "maintain_begin" : "02:00:00",
  "maintain_end" : "06:00:00",
  "enable_log_collection" : false,
  "new_auth_cert" : false,
  "storage_space" : 492,
  "total_storage_space" : 600,
  "used_storage_space" : 25,
  "partition_num" : "300",
  "enable_publicip" : false,
  "ssl_enable" : false,
  "broker_ssl_enable" : false,
  "cert_replaced" : false,
  "kafka_security_protocol" : "SASL_SSL",
  "management_connect_address" : "https://192.168.0.100:9999",
  "cross_vpc_info" : "{\"192.168.0.61\":{\"advertised_ip\":\"192.168.0.61\",\"port\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.72\":{\"advertised_ip\":\"192.168.0.72\",\"port\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"},\"192.168.0.100\":{\"advertised_ip\":\"192.168.0.100\",\"port\":9011,\"port_id\":\"xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx\"}}",
```



```
 "storage_resource_id" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
 "storage_spec_code" : "dms.physical.storage.ultra",
 "service_type" : "advanced",
 "storage_type" : "hec",
 "enterprise_project_id" : "0",
 "retention_policy" : "produce_reject",
 "ipv6_enable" : false,
 "ipv6_connect_addresses" : [ ],
 "rest_enable" : false,
 "rest_connect_address" : "",
 "kafka_public_status" : "closed",
 "public_bandwidth" : 0,
 "trace_enable" : false,
 "agent_enable" : false,
 "pod_connect_address" : "100.86.75.15:9080,100.86.142.77:9080,100.86.250.167:9080",
 "disk_encrypted" : false,
 "kafka_private_connect_address" : "192.168.0.61:9092,192.168.0.100:9092,192.168.0.72:9092",
 "new_spec_billing_enable" : false,
 "ces_version" : "linux",
 "port_protocols" : "{\"private_plain_enable\": true,\"private_plain_address\":
\"192.xxx.xxx.xxx:9092,192.xxx.xxx.xxx:9092,192.xxx.xxx.xxx:9092\",\"private_sasl_ssl_enable\":
true,\"private_sasl_ssl_address\":
\"192.xxx.xxx.xxx:9093,192.xxx.xxx.xxx:9093,192.xxx.xxx.xxx:9093\",\"private_sasl_plaintext_enable\":
false,\"private_sasl_plaintext_address\": \"\",\"public_plain_enable\": true,\"public_plain_address\":
\"100.xxx.xxx.xxx:9094,100.xxx.xxx.xxx:9094,100.xxx.xxx.xxx:9094\",\"public_sasl_ssl_enable\":
true,\"public_sasl_ssl_address\":
\"100.xxx.xxx.xxx:9095,100.xxx.xxx.xxx:9095,100.xxx.xxx.xxx:9095\",\"public_sasl_plaintext_enable\":
false,\"public_sasl_plaintext_address\": \"\"}"
}
```


状态码

状态码	描述
200	查询指定实例成功。

错误码

请参见**错误码**。

7.2.2 实例管理

7.2.2.1 新增 Kafka 实例指定 Topic 分区

功能介绍

新增Kafka实例指定Topic分区。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**修改Kafka实例Topic**。

URI

POST /v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions-reassignment

表 7-55 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	Topic名称。

请求参数

表 7-56 请求 Body 参数

参数	是否必选	参数类型	描述
partition	否	Integer	期望调整分区后的数量，必须大于当前分区数量，小于等于100。

响应参数

无

请求示例

增加Topic分区数。

```
POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/management/topics/{topic}/partitions-reassignment

{
  "partition" : 3
}
```

响应示例

无

状态码

状态码	描述
204	新增分区操作成功。

错误码

请参见**错误码**。

7.2.3 消费组管理

7.2.3.1 重置消费组消费进度到指定位置

功能介绍

Kafka实例不支持在线重置消费进度。在执行重置消费进度之前，必须停止被重置消费组客户端。停止待重置消费组客户端，然后等待一段时间（即ConsumerConfig.SESSION_TIMEOUT_MS_CONFIG配置的时间，默认为1000毫秒）后，服务端才认为此消费组客户端已下线。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**重置消费组消费进度到指定位置**。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/instances/{instance_id}/management/groups/{group}/reset-message-offset

表 7-57 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 7-58 请求 Body 参数

参数	是否必选	参数类型	描述
topic	否	String	topic名称。
partition	是	Integer	分区编号，默认值为-1，若传入值为-1，则重置所有分区。

参数	是否必选	参数类型	描述
message_offset	否	Long	重置消费进度到指定偏移量。 ● 如果传入offset小于当前最小的offset，则重置到最小的offset。 ● 如果大于最大的offset，则重置到最大的offset。 **message_offset、timestamp二者必选其一。**
timestamp	否	Long	重置消费进度到指定时间，格式为unix时间戳，单位为毫秒。 ● 如果传入timestamp早于当前最早的timestamp，则重置到最早的timestamp。 ● 如果晚于最晚的timestamp，则重置到最晚的timestamp。 **message_offset、timestamp二者必选其一。**

响应参数

无

请求示例

- 重置的消费进度到指定偏移量。

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

{
  "topic" : "test",
  "partition" : 0,
  "message_offset" : 10
}
```

- 重置的消费进度到指定时间。

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

{
  "topic" : "test",
  "partition" : 0,
  "timestamp" : 1571812144000
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

- 重置的消费进度到指定偏移量。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetMessageOffsetSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetMessageOffsetRequest request = new ResetMessageOffsetRequest();
        ResetMessageOffsetReq body = new ResetMessageOffsetReq();
        body.withMessageOffset(10L);
        body.withPartition(0);
        body.withTopic("test");
        request.withBody(body);
        try {
            ResetMessageOffsetResponse response = client.resetMessageOffset(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());
        }
    }
}
```

- 重置的消费进度到指定时间。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ResetMessageOffsetSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ResetMessageOffsetRequest request = new ResetMessageOffsetRequest();
        ResetMessageOffsetReq body = new ResetMessageOffsetReq();
        body.withTimestamp(1571812144000L);
        body.withPartition(0);
        body.withTopic("test");
        request.withBody(body);
        try {
            ResetMessageOffsetResponse response = client.resetMessageOffset(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

- 重置的消费进度到指定偏移量。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.getenv("CLOUD_SDK_AK")
    sk = os.getenv("CLOUD_SDK_SK")
```

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

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

    try:
        request = ResetMessageOffsetRequest()
        request.body = ResetMessageOffsetReq(
            message_offset=10,
            partition=0,
            topic="test"
        )
        response = client.reset_message_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)
```

- 重置的消费进度到指定时间。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.getenv("CLOUD_SDK_AK")
    sk = os.getenv("CLOUD_SDK_SK")

    credentials = BasicCredentials(ak, sk) \

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

    try:
        request = ResetMessageOffsetRequest()
        request.body = ResetMessageOffsetReq(
            timestamp=1571812144000,
            partition=0,
            topic="test"
        )
        response = client.reset_message_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

- 重置的消费进度到指定偏移量。

```
package main

import (
```

```
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetMessageOffsetRequest{}
    messageOffsetResetMessageOffsetReq:= int64(10)
    topicResetMessageOffsetReq:= "test"
    request.Body = &model.ResetMessageOffsetReq{
        MessageOffset: &messageOffsetResetMessageOffsetReq,
        Partition: int32(0),
        Topic: &topicResetMessageOffsetReq,
    }
    response, err := client.ResetMessageOffset(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

- 重置的消费进度到指定时间。

```
package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResetMessageOffsetRequest{}
    timestampResetMessageOffsetReq:= int64(1571812144000)
    topicResetMessageOffsetReq:= "test"
    request.Body = &model.ResetMessageOffsetReq{
        Timestamp: &timestampResetMessageOffsetReq,
        Partition: int32(0),
        Topic: &topicResetMessageOffsetReq,
    }
    response, err := client.ResetMessageOffset(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

状态码

状态码	描述
204	重置消费组消息进度到指定位置操作成功。

错误码

请参见**错误码**。

7.2.3.2 查询所有消费组

功能介绍

查询所有消费组。

调用方法

请参见**如何调用API**。

URI

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

表 7-59 路径参数

参数	是否必选	参数类型	描述
engine	是	String	引擎。
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 7-60 Query 参数

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。
limit	否	Integer	当次查询返回的最大消费组id个数，默认值为10，取值范围为1~50。
group	否	String	消费组名过滤查询，过滤方式为字段包含过滤。
topic	否	String	若指定topic，则只查询该topic的消费组。

请求参数

无

响应参数

状态码： 200

表 7-61 响应 Body 参数

参数	参数类型	描述
groups	Array of **GroupInfoSimple** objects	所有的消费组。
total	Integer	所有的消费组总数。

表 7-62 GroupInfoSimple

参数	参数类型	描述
createdAt	Long	创建时间。
group_id	String	消费组ID。

参数	参数类型	描述
state	String	消费组状态。包含以下状态： ● Dead：消费组内没有任何成员，且没有任何元数据。 ● Empty：消费组内没有任何成员，存在元数据。 ● PreparingRebalance：准备开启rebalance。 ● CompletingRebalance：所有成员加入group。 ● Stable：消费组内成员可正常消费。
coordinator_id	Integer	协调器编号。
group_desc	String	消费组描述。
lag	Long	堆积数。

请求示例

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

响应示例

状态码： 200

查询实例集群的所有消费组成功。

```
{
  "groups" : [ {
    "createdAt" : 1691401194847,
    "group_id" : "consumer-1",
    "state" : "EMPTY",
    "coordinator_id" : 1,
    "lag" : 0,
    "group_desc" : null
  }, {
    "createdAt" : 1691401194960,
    "group_id" : "consumer-2",
    "state" : "STABLE",
    "coordinator_id" : 2,
    "lag" : 0,
    "group_desc" : null
  }, {
    "createdAt" : 1691401207309,
    "group_id" : "consumer-3",
    "state" : "STABLE",
    "coordinator_id" : 3,
    "lag" : 0,
    "group_desc" : null
  } ],
  "total" : 3
}
```

状态码

状态码	描述
200	查询实例集群的所有消费组成功。

错误码

请参见**错误码**。

7.2.4 Smart Connect

7.2.4.1 修改转储任务的配额

功能介绍

修改转储任务的配额。

2022年9月前创建的实例支持调用此接口新增转储任务配额，2022年9月及以后创建的实例，转储任务配额默认为最大值，由于转储任务配额不支持减少，调用此接口修改转储任务配额会报错。

调用方法

请参见**如何调用API**。

URI

PUT /v2/{project_id}/connectors/{connector_id}/sink-tasks

表 7-63 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**。
connector_id	是	String	实例转储ID。 请参考**查询实例**返回的数据。

请求参数

表 7-64 请求 Body 参数

参数	是否必选	参数类型	描述
sink_max_tasks	是	Integer	转储任务的总个数。

响应参数

无

请求示例

修改转储任务的配额。

```
PUT https://{endpoint}/v2/{project_id}/connectors/{connector_id}/sink-tasks

{
  "sink_max_tasks" : 9
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

修改转储任务的配额。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class UpdateSinkTaskQuotaSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        UpdateSinkTaskQuotaRequest request = new UpdateSinkTaskQuotaRequest();
        UpdateSinkTaskQuotaReq body = new UpdateSinkTaskQuotaReq();
        body.withSinkMaxTasks(9);
        request.withBody(body);
```

```
        try {
            UpdateSinkTaskQuotaResponse response = client.updateSinkTaskQuota(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

修改转储任务的配额。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = os.getenv("CLOUD_SDK_AK")
    sk = os.getenv("CLOUD_SDK_SK")

    credentials = BasicCredentials(ak, sk) \

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

    try:
        request = UpdateSinkTaskQuotaRequest()
        request.body = UpdateSinkTaskQuotaReq(
            sink_max_tasks=9
        )
        response = client.update_sink_task_quota(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

修改转储任务的配额。

```
package main

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

```
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateSinkTaskQuotaRequest{}
    request.Body = &model.UpdateSinkTaskQuotaReq{
        SinkMaxTasks: int32(9),
    }
    response, err := client.UpdateSinkTaskQuota(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

状态码

状态码	描述
204	修改转储任务配额成功。

错误码

请参见**错误码**。

7.2.4.2 创建关闭实例转储节点的订单

功能介绍

创建关闭实例转储节点的订单。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**关闭Smart Connect（按需实例）**。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/kafka/instances/{instance_id}/delete-connector-order

表 7-65 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**。 最小长度：**1** 最大长度：**100**
instance_id	是	String	实例ID。 最小长度：**1** 最大长度：**100**

请求参数

表 7-66 请求 Body 参数

参数	是否必选	参数类型	描述
instance_id	是	String	需要关闭connector的实例id，和请求路径上的一致。
url	否	String	提交关闭connector订单后前端跳转的页面

响应参数

状态码： 200

表 7-67 响应 Body 参数

参数	参数类型	描述
order_id	String	返回cbc生成的订单id。

请求示例

```
POST https://{endpoint}/v2/{project_id}/kafka/instances/{instance_id}/delete-connector-order

{
  "instance_id" : "20c6b355-5d95-45ef-b476-e38bcccce0d7",
  "url" : "https://console.xxx.xxx.com/dms/?engine=all&region=xxx&locale=xxx#/queue/manager/newKafkaList"
}
```

响应示例

状态码： 200

创建关闭实例转储节点的订单成功

```
{
  "order_id" : "CS2304180947HLABU"
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateDeleteConnectorOrderSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateDeleteConnectorOrderRequest request = new CreateDeleteConnectorOrderRequest();
        ConnectorOrderRequestBody body = new ConnectorOrderRequestBody();
        body.withUrl("https://console.xxx.xxx.com/dms/?engine=all&region=xxx&locale=xxx#/queue/manager/newKafkaList");
        body.withInstanceId("20c6b355-5d95-45ef-b476-e38bcccce0d7");
        request.withBody(body);
        try {
            CreateDeleteConnectorOrderResponse response = client.createDeleteConnectorOrder(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 huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateDeleteConnectorOrderRequest()
        request.body = ConnectorOrderRequestBody(
            url="https://console.xxx.xxx.com/dms/?engine=all&region=xxx&locale=xxx#/queue/manager/newKafkaList",
            instance_id="20c6b355-5d95-45ef-b476-e38bcccce0d7"
        )
        response = client.create_delete_connector_order(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateDeleteConnectorOrderRequest{}
    urlConnectorOrderRequestBody:= "https://console.xxx.xxx.com/dms/?engine=all&region=xxx&locale=xxx#/
queue/manager/newKafkaList"
    request.Body = &model.ConnectorOrderRequestBody{
        Url: &urlConnectorOrderRequestBody,
        InstanceId: "20c6b355-5d95-45ef-b476-e38bcccce0d7",
    }
    response, err := client.CreateDeleteConnectorOrder(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

状态码

状态码	描述
200	创建关闭实例转储节点的订单成功

错误码

请参见**错误码**。

7.2.4.3 创建转储任务

功能介绍

创建转储任务。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**开启Smart Connect（按需实例）**。

调用方法

请参见**如何调用API**。

URI

POST /v2/{project_id}/connectors/{connector_id}/sink-tasks

表 7-68 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**。

参数	是否必选	参数类型	描述
connector_id	是	String	实例转储ID。 请参考**查询实例**返回的数据。

请求参数

表 7-69 请求 Body 参数

参数	是否必选	参数类型	描述
source_type	是	String	源数据类型，目前只支持BLOB。
task_name	是	String	转储任务名称。
destination_type	是	String	转存的目标类型，当前只支持OBS。
obs_destination_descriptor	是	**ObsDestinationDescriptor** object	转存目标的描述。

表 7-70 ObsDestinationDescriptor

参数	是否必选	参数类型	描述
topics	是	String	转存的topic列表名称，支持输入多个topic，以逗号“,”分隔。同时支持正则表达式。
topics_regex	否	String	转存topic的正则表达式，与topics必须二选一，不能同时都设置或者“.*”。
consumer_strategy	是	String	转储启动偏移量： • latest: 从Topic最后端开始消费。 • earliest: 从Topic最前端消息开始消费。 默认是latest。
destination_file_type	是	String	转储文件格式。当前只支持text。
access_key	是	String	访问密钥AK。
secret_key	是	String	访问密钥SK。

参数	是否必选	参数类型	描述
obs_bucket_name	是	String	存储该通道数据的OBS桶名称。
obs_path	否	String	存储在obs的路径，默认可以不填。 取值范围：英文字母、数字、下划线、中划线和斜杠，最大长度为64个字符。 默认配置为空。
partition_format	是	String	将转储文件的生成时间使用“yyyy/MM/dd/HH/mm”格式生成分区字符串，用来定义写到OBS的Object文件所在的目录层次结构。 ● yyyy：年 ● yyyy/MM：年/月 ● yyyy/MM/dd：年/月/日 ● yyyy/MM/dd/HH：年/月/日/时 ● yyyy/MM/dd/HH/mm：年/月/日/时/分，例如：2017/11/10/14/49，目录结构就是“2017 > 11 > 10 > 14 > 49”，“2017”表示最外层文件夹。 **说明** 数据转储成功后，存储的目录结构为“obs_bucket_path/file_prefix/partition_format”。默认时间是GMT+8 时间
record_delimiter	否	String	转储文件的记录分隔符，用于分隔写入转储文件的用户数据。 取值范围： ● 逗号“,” ● 分号“;” ● 竖线“|” ● 换行符“\n” ● NULL 默认值：换行符“\n”。

参数	是否必选	参数类型	描述
deliver_time_interval	是	Integer	根据用户配置的时间，周期性的将数据导入OBS，若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30～900 单位：秒。 **说明** 使用OBS通道转储流式数据时该参数为必选配置。

响应参数

状态码： 200

表 7-71 响应 Body 参数

参数	参数类型	描述
task_id	String	任务ID。

请求示例

创建一个转储任务，转储topic-test中的数据到OBS。

```
POST https://{endpoint}/v2/{project_id}/connectors/{connector_id}/sink-tasks

{
  "source_type" : "BLOB",
  "task_name" : "obsTransfer-1122976956",
  "destination_type" : "OBS",
  "obs_destination_descriptor" : {
    "consumer_strategy" : "earliest",
    "destination_file_type" : "TEXT",
    "access_key" : "XXXXXXXXXXXXXXXXXXX",
    "secret_key" : "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "obs_bucket_name" : "6666",
    "obs_path" : "obsTransfer-1122976956",
    "partition_format" : "yyyy/MM/dd/HH/mm",
    "record_delimiter" : "",
    "deliver_time_interval" : 300,
    "topics" : "topic-test"
  }
}
```

响应示例

状态码： 200

创建转储任务成功。

```
{
  "task_id" : "2962882a-386c-4c9d-bb59-3b4f55d82961"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建一个转储任务，转储topic-test中的数据到OBS。

```
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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class CreateSinkTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        CreateSinkTaskRequest request = new CreateSinkTaskRequest();
        CreateSinkTaskReq body = new CreateSinkTaskReq();
        ObsDestinationDescriptor obsDestinationDescriptorbody = new ObsDestinationDescriptor();
        obsDestinationDescriptorbody.withTopics("topic-test")
            .withConsumerStrategy(ObsDestinationDescriptor.ConsumerStrategyEnum.fromValue("earliest"))
            .withDestinationFileType(ObsDestinationDescriptor.DestinationFileTypeEnum.fromValue("TEXT"))
            .withAccessKey("XXXXXXXXXXXXXXXXXXXX")
            .withSecretKey("XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX")
            .withObsBucketName("6666")
            .withObsPath("obsTransfer-1122976956")
            .withPartitionFormat(ObsDestinationDescriptor.PartitionFormatEnum.fromValue("yyyy/MM/dd/HH/mm"))
            .withRecordDelimiter("")
            .withDeliverTimeInterval(300);
        body.withObsDestinationDescriptor(obsDestinationDescriptorbody);
        body.withDestinationType(CreateSinkTaskReq.DestinationTypeEnum.fromValue("OBS"));
        body.withTaskName("obsTransfer-1122976956");
        body.withSourceType(CreateSinkTaskReq.SourceTypeEnum.fromValue("BLOB"));
        request.withBody(body);
        try {
            CreateSinkTaskResponse response = client.createSinkTask(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

创建一个转储任务，转储topic-test中的数据到OBS。

```
# coding: utf-8

from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateSinkTaskRequest()
        obsDestinationDescriptorbody = ObsDestinationDescriptor(
            topics="topic-test",
            consumer_strategy="earliest",
            destination_file_type="TEXT",
            access_key="XXXXXXXXXXXXXXXXXXX",
            secret_key="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
            obs_bucket_name="6666",
            obs_path="obsTransfer-1122976956",
            partition_format="yyyy/MM/dd/HH/mm",
            record_delimiter="",
            deliver_time_interval=300
        )
        request.body = CreateSinkTaskReq(
            obs_destination_descriptor=obsDestinationDescriptorbody,
            destination_type="OBS",
            task_name="obsTransfer-1122976956",
            source_type="BLOB"
        )
        response = client.create_sink_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

创建一个转储任务，转储topic-test中的数据到OBS。

```
package main
```

```
import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.CreateSinkTaskRequest{}
    obsPathObsDestinationDescriptor:= "obsTransfer-1122976956"
    recordDelimiterObsDestinationDescriptor:= ""
    obsDestinationDescriptorbody := &model.ObsDestinationDescriptor{
        Topics: "topic-test",
        ConsumerStrategy: model.GetObsDestinationDescriptorConsumerStrategyEnum().EARLIEST,
        DestinationFileType: model.GetObsDestinationDescriptorDestinationFileTypeEnum().TEXT,
        AccessKey: "XXXXXXXXXXXXXXXXXXXX",
        SecretKey: "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
        ObsBucketName: "6666",
        ObsPath: &obsPathObsDestinationDescriptor,
        PartitionFormat: model.GetObsDestinationDescriptorPartitionFormatEnum().YYYY_MM_DD_HH_MM,
        RecordDelimiter: &recordDelimiterObsDestinationDescriptor,
        DeliverTimeInterval: int32(300),
    }
    request.Body = &model.CreateSinkTaskReq{
        ObsDestinationDescriptor: obsDestinationDescriptorbody,
        DestinationType: model.GetCreateSinkTaskReqDestinationTypeEnum().OBS,
        TaskName: "obsTransfer-1122976956",
        SourceType: model.GetCreateSinkTaskReqSourceTypeEnum().BLOB,
    }
    response, err := client.CreateSinkTask(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

状态码

状态码	描述
200	创建转储任务成功。

错误码

请参见**错误码**。

7.2.4.4 查询转储任务列表

功能介绍

查询转储任务列表。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**查询Smart Connect任务列表**。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/connectors/{connector_id}/sink-tasks

表 7-72 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**。
connector_id	是	String	实例转储ID。 请参考**查询实例**返回的数据。

请求参数

无

响应参数

状态码： 200

表 7-73 响应 Body 参数

参数	参数类型	描述
tasks	Array of **tasks** objects	转储任务列表。
total_number	Integer	转储任务总数。
max_tasks	Integer	总的支持任务个数。
quota_tasks	Integer	任务总数的配额。

表 7-74 tasks

参数	参数类型	描述
task_id	String	任务ID。
task_name	String	转储任务名称。
destination_type	String	转储任务类型。
create_time	Long	转储任务创建时间戳。
status	String	转储任务状态。
topics	String	返回任务转存的topics列表或者正则表达式。

请求示例

查询转储任务列表。

```
GET https://{endpoint}/v2/{project_id}/connectors/{connector_id}/sink-tasks
```

响应示例

状态码： 200

查询转储任务列表成功。

```
{
  "tasks" : [ {
    "task_id" : "2e148bed-3038-4617-8ade-b52e84a33eeb",
    "task_name" : "obsTransfer-1122976956",
    "destination_type" : "OBS",
    "create_time" : 1592309487621,
    "status" : "RUNNING",
    "topics" : "topic-test"
  } ],
  "total_number" : 1,
  "max_tasks" : 9,
  "quota_tasks" : 10
}
```

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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
```

```
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ListSinkTasksSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ListSinkTasksRequest request = new ListSinkTasksRequest();
        try {
            ListSinkTasksResponse response = client.listSinkTasks(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 huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListSinkTasksRequest()
        response = client.list_sink_tasks(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

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

状态码

状态码	描述
200	查询转储任务列表成功。

错误码

请参见**错误码**。

7.2.4.5 查询单个转储任务

功能介绍

查询单个转储任务。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**查询Smart Connector任务详情**。

调用方法

请参见**如何调用API**。

URI

GET /v2/{project_id}/connectors/{connector_id}/sink-tasks/{task_id}

表 7-75 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**。
connector_id	是	String	实例转储ID。 请参考**查询实例**返回的数据。
task_id	是	String	转储任务ID。

表 7-76 Query 参数

参数	是否必选	参数类型	描述
topic-info	否	String	是否包含topic信息。默认是false。

请求参数

无

响应参数

状态码： 200

表 7-77 响应 Body 参数

参数	参数类型	描述
task_name	String	转储任务名称。
destination_type	String	转储任务类型。
create_time	Long	转储任务创建时间戳。
status	String	转储任务状态。

参数	参数类型	描述
topics	String	返回任务转存的topics列表或者正则表达式。
obs_destination_descriptor	obs_destination_descriptor object	转存目标的描述。
topics_info	Array of topics_info objects	topic信息。

表 7-78 obs_destination_descriptor

参数	参数类型	描述
consumer_strategy	String	消费启动策略： • latest：从Topic最后端开始消费。 • earliest: 从Topic最前端消息开始消费。 默认是latest。
destination_file_type	String	转储文件格式。目前只支持text格式。
obs_bucket_name	String	存储该通道数据的OBS桶名称。
obs_path	String	存储在obs的路径。
partition_format	String	将转储文件的生成时间使用“yyyy/MM/dd/HH/mm”格式生成分区字符串，用来定义写到OBS的Object文件所在的目录层次结构。 • yyyy：年 • yyyy/MM：年/月 • yyyy/MM/dd：年/月/日 • yyyy/MM/dd/HH：年/月/日/时 • yyyy/MM/dd/HH/mm：年/月/日/时/分，例如：2017/11/10/14/49，目录结构就是“2017 > 11 > 10 > 14 > 49”，“2017”表示最外层文件夹。 **说明** 数据转储成功后，存储的目录结构为“obs_bucket_path/file_prefix/partition_format”。默认时间是GMT+8 时间

参数	参数类型	描述
record_delimiter	String	转储文件的记录分隔符，用于分隔写入转储文件的用户数据。 取值范围： ● 逗号“,” ● 分号“;” ● 竖线“|” ● 换行符“\n” ● NULL 默认值：换行符“\n”。
deliver_time_interval	Integer	根据用户配置的时间，周期性的将数据导入OBS，若某个时间段内无数据，则此时间段不会生成打包文件。 取值范围：30～900，缺省值：300，单位：秒。 **说明** 使用OBS通道转储流式数据时该参数为必选配置。
obs_part_size	Long	每个传输文件多大后就开始上传，单位为byte。 默认值5242880。

表 7-79 topics_info

参数	参数类型	描述
topic	String	topic名称。
partitions	Array of **partitions** objects	分区列表。

表 7-80 partitions

参数	参数类型	描述
partition_id	String	分区ID。
status	String	运行状态。
last_transfer_offset	String	已转储的消息偏移量。
log_end_offset	String	消息偏移量。
lag	String	积压的消息数。

请求示例

查询指定的转储任务详情。

```
GET https://{endpoint}/v2/{project_id}/connectors/{connector_id}/sink-tasks/{task_id}?topic-info=true
```

响应示例

状态码： 200

查询单个转储任务成功。

```
{
  "task_name" : "obsTransfer-56997523",
  "destination_type" : "OBS",
  "create_time" : 1628126621283,
  "status" : "RUNNING",
  "topics" : "topic-sdk-no-delete",
  "obs_destination_descriptor" : {
    "consumer_strategy" : "earliest",
    "destination_file_type" : "TEXT",
    "obs_bucket_name" : "testobs",
    "obs_path" : "obsTransfer-56997523",
    "partition_format" : "yyyy/MM/dd/HH/mm",
    "record_delimiter" : "",
    "deliver_time_interval" : 300,
    "obs_part_size" : 5242880
  },
  "topics_info" : [ {
    "topic" : "topic-sdk-no-delete",
    "partitions" : [ {
      "partition_id" : "2",
      "status" : "RUNNING",
      "last_transfer_offset" : "3",
      "log_end_offset" : "3",
      "lag" : "0"
    }, {
      "partition_id" : "1",
      "status" : "RUNNING",
      "last_transfer_offset" : "3",
      "log_end_offset" : "3",
      "lag" : "0"
    }, {
      "partition_id" : "0",
      "status" : "RUNNING",
      "last_transfer_offset" : "3",
      "log_end_offset" : "3",
      "lag" : "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.kafka.v2.region.KafkaRegion;
```

```
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class ShowSinkTaskDetailSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        ShowSinkTaskDetailRequest request = new ShowSinkTaskDetailRequest();
        request.withTopicInfo(ShowSinkTaskDetailRequest.TopicInfoEnum.fromValue("<topic-info>"));
        try {
            ShowSinkTaskDetailResponse response = client.showSinkTaskDetail(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 huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowSinkTaskDetailRequest()
```

```
        request.topic_info = "<topic-info>"
        response = client.show_sink_task_detail(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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowSinkTaskDetailRequest{}
    topicInfoRequest:= model.GetShowSinkTaskDetailRequestTopicInfoEnum().<TOPIC_INFO>
    request.TopicInfo = &topicInfoRequest
    response, err := client.ShowSinkTaskDetail(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}
```

状态码

状态码	描述
200	查询单个转储任务成功。

错误码

请参见**错误码**。

7.2.4.6 删除单个转储任务

功能介绍

删除单个转储任务。

说明

当前页面API为历史版本API，未来可能停止维护。请使用**删除Smart Connector任务**。

调用方法

请参见**如何调用API**。

URI

DELETE /v2/{project_id}/connectors/{connector_id}/sink-tasks/{task_id}

表 7-81 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**。
connector_id	是	String	实例转储ID。 请参考**查询实例**返回的数据。
task_id	是	String	转储任务ID。

请求参数

无

响应参数

无

请求示例

删除指定的转储任务。

```
DELETE https://{endpoint}/v2/{project_id}/connectors/{connector_id}/sink-tasks/{task_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.kafka.v2.region.KafkaRegion;
import com.huaweicloud.sdk.kafka.v2.*;
import com.huaweicloud.sdk.kafka.v2.model.*;


public class DeleteSinkTaskSolution {

    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);

        KafkaClient client = KafkaClient.newBuilder()
                .withCredential(auth)
                .withRegion(KafkaRegion.valueOf("<YOUR REGION>"))
                .build();
        DeleteSinkTaskRequest request = new DeleteSinkTaskRequest();
        try {
            DeleteSinkTaskResponse response = client.deleteSinkTask(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 huaweicloudsdkkafka.v2.region.kafka_region import KafkaRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkkafka.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 = KafkaClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(KafkaRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteSinkTaskRequest()
        response = client.delete_sink_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"
    kafka "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/kafka/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 := kafka.NewKafkaClient(
        kafka.KafkaClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

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

状态码

状态码	描述
204	删除转储任务成功。

错误码

请参见**错误码**。

8 附录

8.1 状态码

状态码如表8-1所示

表 8-1 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。

8.2 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.00400077	Insufficient IPs in the selected subnet.	所选择的子网中可用ip数量少于所需ip数量。	选择其他ip数量充足的子网。
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

状态码	错误码	错误信息	描述	处理措施
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
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 1-200.	Kafka实例创建Topic的分区数必须在1-200范围内。	请检查topic分区数

状态码	错误码	错误信息	描述	处理措施
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
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 scale 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 scale 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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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。
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

状态码	错误码	错误信息	描述	处理措施
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.00500033	Failed to access EPS to update the project	访问EPS更新project失败	请联系技术支持工程师协助解决。
400	DMS.00500960	Invalid user AK/SK.	用户AK SK非法	请核实用户AK SK后重试。
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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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
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.111400860	Instance partition is not enough. Total partition is over the partition limitation.	实例分区配额不足，超出实例分区上限	请检查分区数是否超过限制。
400	DMS.40001016	Kafka Connector配置错误	Kafka Connector配置错误	根据错误提示信息检查配置并修复
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

状态码	错误码	错误信息	描述	处理措施
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
403	DMS.00403007	Authorization denied.	用户缺少权限，无法执行操作。	请检查用户权限
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

状态码	错误码	错误信息	描述	处理措施
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
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

状态码	错误码	错误信息	描述	处理措施
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
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。

状态码	错误码	错误信息	描述	处理措施
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。
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。

状态码	错误码	错误信息	描述	处理措施
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
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

8.3 实例状态说明

表 8-2 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。

状态	说明
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。

8.4 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 8-1 查看项目 ID

----结束

8.5 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 8-2 查看账号名和账号 ID

A 修订记录

发布日期	修订记录
2025-02-11	本次变更如下： • 在**创建实例**中，新增“port_protocol”参数。
2024-12-24	本次变更如下： • 新增**修改Kafka的接入方式**、**查询消费组消息位点**、**修改所有消费组**、**查询指定消费组**、**删除指定消费组**、**查询指定消费组的topic**、**查询指定消费组的消费成员**、**Kafka删除消息**和**查询Kafka产品规格核数**接口。
2024-06-04	本次变更如下： • 新增**诊断管理**相关接口。
2024-03-06	本次变更如下： • 新增**Smart Connect**相关接口。 • 将转储相关接口移动至**历史API**。
2024-02-26	本次变更如下： • 新增**生产消息**接口。
2023-11-10	本次变更如下： • 新增**获取实例配置**和**修改实例配置**接口。
2023-07-26	本次变更如下： • 新增**查询Topic的分区列表**和**查询Topic的当前生产者列表**接口。
2023-06-06	本次变更如下： • 新增**创建消费组**接口。
2023-04-23	本次变更如下： • 新增**Kafka实例批量删除Group**接口。

发布日期	修订记录
2023-02-03	本次变更如下： ● 更新**创建实例**、**实例规格变更**和**查询实例的扩容规格列表**的URI，支持实例新规格。
2022-06-30	本次变更如下： ● 在**创建实例**的“specification”参数中，增加新规格描述。 ● 在**创建实例**中，新增“broker_num”参数。
2021-12-14	本次变更如下： ● 在**权限和授权项**中，将授权项明细接口从V1改为V2。
2021-11-16	本次变更如下： ● 增加V2接口API。
2020-10-13	第一次正式发布。