

分布式消息服务 RabbitMQ 版

API 参考

文档版本 12
发布日期 2025-02-14



版权所有 © 华为云计算技术有限公司 2025。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

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

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

目录

1 使用前必读	1
2 API 概览	3
3 如何调用 API	5
3.1 构造请求	5
3.2 认证鉴权	8
3.3 返回结果	10
4 快速入门	12
5 API V2 (推荐)	14
5.1 生命周期管理	14
5.1.1 创建实例	14
5.1.2 查询所有实例列表	32
5.1.3 查询指定实例	41
5.1.4 删除指定的实例	50
5.1.5 修改实例信息	53
5.1.6 批量删除实例	63
5.1.7 开启 RabbitMQ 实例域名访问能力	70
5.2 实例管理	73
5.2.1 重置密码	73
5.2.2 查询插件列表	77
5.2.3 开启或关闭插件	81
5.3 规格变更管理	85
5.3.1 查询新规格可扩容规格列表	85
5.3.2 新规格实例的规格变更	92
5.4 Vhost 管理	108
5.4.1 创建 Vhost	109
5.4.2 查询 Vhost 列表	112
5.4.3 批量删除指定 Vhost	117
5.5 Exchange 管理	120
5.5.1 创建 Exchange	120
5.5.2 查询 Exchange 列表	125
5.5.3 批量删除指定 Exchange	129
5.6 Queue 管理	133

5.6.1 创建 Queue.....	133
5.6.2 查询所属 Vhost 下 Queue 的列表.....	139
5.6.3 批量删除指定 Queue.....	143
5.6.4 清空 Queue 消息.....	147
5.6.5 查询指定 Queue 详情.....	151
5.7 Binding 管理.....	156
5.7.1 添加绑定.....	156
5.7.2 查询 Exchange 绑定信息列表.....	160
5.7.3 删除绑定.....	165
5.8 用户管理.....	168
5.8.1 创建用户.....	168
5.8.2 查询用户列表.....	175
5.8.3 修改用户参数.....	180
5.8.4 删除用户.....	186
5.9 后台任务管理.....	189
5.9.1 查询实例的后台任务列表.....	190
5.9.2 查询后台任务管理中的指定记录.....	194
5.9.3 删除后台任务管理中的指定记录.....	198
5.10 标签管理.....	201
5.10.1 批量添加或删除实例标签.....	201
5.10.2 查询实例标签.....	206
5.10.3 查询项目标签.....	210
5.11 其他接口.....	214
5.11.1 查询维护时间窗时间段.....	214
5.11.2 查询可用区信息.....	218
5.11.3 查询产品规格列表.....	222
5.11.4 查询实例在 CES 的监控层级关系.....	228
5.11.5 查询 RabbitMQ 产品规格核数.....	234
6 权限和授权项.....	236
7 历史 API.....	239
7.1 API V1.....	239
7.1.1 实例管理类接口.....	239
7.1.1.1 创建实例.....	239
7.1.1.2 查询指定实例.....	242
7.1.1.3 修改实例信息.....	247
7.1.1.4 删除指定实例.....	250
7.1.1.5 批量删除实例.....	251
7.1.1.6 查询所有实例列表.....	253
7.1.2 其他接口.....	258
7.1.2.1 查询可用区信息.....	258
7.1.2.2 查询产品规格列表.....	260
7.1.2.3 查询维护时间窗时间段.....	271

7.1.2.4 查询配额.....	273
7.1.2.5 查询实例在 CES 的监控层级关系.....	275
7.2 API V2.....	278
7.2.1 查询产品规格列表.....	278
7.2.2 RabbitMQ 实例内核升级（废弃）.....	284
7.2.3 查询指定实例.....	285
8 附录.....	292
8.1 状态码.....	292
8.2 错误码.....	294
8.3 实例状态说明.....	315
8.4 获取项目 ID.....	316
8.5 获取账号名和账号 ID.....	317
A 修订记录.....	318

1 使用前必读

欢迎使用分布式消息服务RabbitMQ版。分布式消息服务RabbitMQ版是一项基于高可用分布式集群技术的消息中间件服务，提供了可靠且可扩展的托管消息队列，用于收发消息和存储消息。

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

分布式消息服务RabbitMQ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。

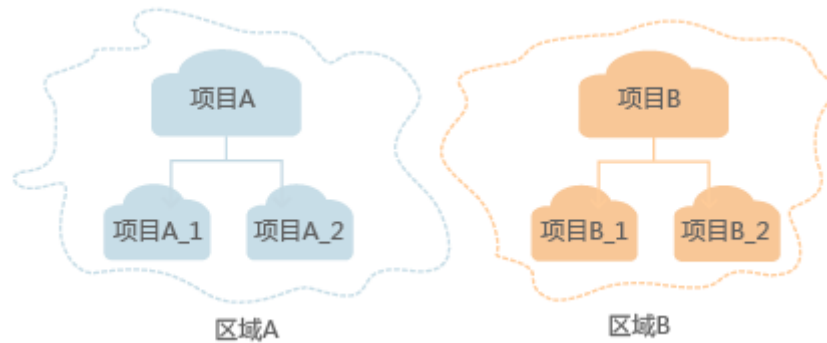
基本概念

- **账号**
用户注册账号时，账号对其所拥有的资源及云服务具有完全的访问权限，可以重置用户密码、分配用户权限等。由于账号是付费主体，为了确保账号安全，建议您不要直接使用账号进行日常管理工作，而是创建用户并使用他们进行日常工作。
- **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）。
在[我的凭证](#)下，您可以查看账号ID和用户ID。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，并在子项目中购买资源，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



同样在[我的凭证](#)下，您可以查看项目ID。

- 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： <ul style="list-style-type: none">• 创建实例• 查询所有实例列表• 查询指定实例• 删除指定实例• 修改实例信息• 批量删除实例
实例管理	包括： <ul style="list-style-type: none">• 重置密码• 查询插件列表• 开启或关闭插件
规格变更管理	包括： <ul style="list-style-type: none">• 查询新规格可扩容规格列表• 新规格实例的规格变更
Vhost管理	包括： <ul style="list-style-type: none">• 创建Vhost• 查询Vhost列表• 批量删除指定Vhost
Exchange管理	包括： <ul style="list-style-type: none">• 创建Exchange• 查询Exchange列表• 批量删除指定Exchange

API	说明
Queue管理	包括： <ul style="list-style-type: none"> • 创建Queue • 查询所属Vhost下Queue的列表 • 批量删除指定Queue • 清空Queue消息 • 查询指定Queue详情
Binding管理	包括： <ul style="list-style-type: none"> • 添加绑定 • 查询Exchange绑定信息列表 • 删除绑定
用户管理	包括： <ul style="list-style-type: none"> • 创建用户 • 查询用户列表 • 修改用户参数 • 删除用户
后台任务管理	包括： <ul style="list-style-type: none"> • 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： <ul style="list-style-type: none"> • 批量添加或删除实例标签 • 查询实例标签 • 查询项目标签
其他接口	包括： <ul style="list-style-type: none"> • 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RabbitMQ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。

您还可以通过这个视频教程了解如何构造请求调用API：<https://bbs.huaweicloud.com/videos/102987>。

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“华北-北京四”区域的Endpoint为“iam.cn-north-4.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“华北-北京四”区域的Endpoint：“iam.cn-north-4.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.cn-north-4.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.cn-north-4.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。	e9993fc787d94b6c886cb aa340f9c0f4
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。 MIIPAgYJKoZlhvcNAQcCo ...ggg1BBIIlNPXsidG9rZ

 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.cn-north-4.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.cn-north-4.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": "xxxxxxx" //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.cn-north-4.myhuaweicloud.com/v3.0/OS-USER/users
Content-Type: application/json
X-Auth-Token: ABCDEFJ....
```

您可以通过这个视频教程了解如何使用Token认证：<https://bbs.huaweicloud.com/videos/101333>。

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_ 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RabbitMQ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RabbitMQ的Endpoint，具体请参见[地区和终端节点](#)。

创建 RabbitMQ 实例

如下示例是创建RabbitMQ实例的请求消息：

```
{
  "name": "rabbitmq",
  "engine": "rabbitmq",
  "engine_version": "3.8.35",
  "storage_space": 100,
  "access_user": "test",
  "password": "ZxxxA",
  "vpc_id": "eadxxe72c",
  "security_group_id": "aa75axxc8c73220",
  "subnet_id": "3cb6axxx671d6a8",
  "available_zones": [
    "effdcxxb42f56533"
  ],
  "product_id": "c6.2u4g.single",
  "storage_spec_code": "dms.physical.storage.ultra.v2"
}
```

- name：实例名称。由您自行定义。
- engine：消息引擎，设置rabbitmq。
- engine_version：消息引擎的版本。
- storage_space：消息存储空间，单位GB。具体取值范围，请参考[创建实例](#)。
- access_user：登录RabbitMQ的用户名，由您自行定义。
- password：登录RabbitMQ的密码，由您自行定义。
- vpc_id：RabbitMQ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：rabbitmq。
engine_version	是	String	消息引擎的版本。 <ul style="list-style-type: none"> RabbitMQ版本有：3.8.35、AMQP-0-9-1。
enable_acl	否	Boolean	ACL访问控制（仅AMQP版本支持此参数）
storage_space	是	Integer	消息存储空间，单位GB。 <ul style="list-style-type: none"> 单机RabbitMQ实例的存储空间的取值范围 100GB~90000GB。 集群RabbitMQ实例的存储空间的取值范围为100GB*节点数~90000GB、200GB*节点数~90000GB、300GB*节点数~90000GB。
access_user	否	String	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。

参数	是否必选	参数类型	描述
password	否	String	实例的认证密码。 复杂度要求： <ul style="list-style-type: none"> • 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两种组合： <ul style="list-style-type: none"> - 小写字母 - 大写字母 - 数字 - 特殊字符包括 (`~!@#\$%^&*()-_+=+ [{}]:",<.>/?)
vpc_id	是	String	租户VPC 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	租户安全组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。
subnet_id	是	String	子网ID。 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。
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。
product_id	是	String	产品标识。 产品ID可以从 查询产品规格列表 获取。 如果产品ID为集群类型（即对应的type为cluster），broker_num字段为必选。

参数	是否必选	参数类型	描述
broker_num	否	Integer	代理个数。 当产品为单机类型，代理个数只能为1；当产品为集群类型，可选3、5、7个代理个数。 产品类型单为single时： <ul style="list-style-type: none"> • 1 产品类型单为cluster时： <ul style="list-style-type: none"> • 3 • 5 • 7
maintain_begin	否	String	维护时间窗开始时间，格式为HH:mm。 <ul style="list-style-type: none"> • 维护时间窗开始和结束时间必须为指定的时间段。 • 开始时间必须为22:00、02:00、06:00、10:00、14:00和18:00。 • 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。
maintain_end	否	String	维护时间窗结束时间，格式为HH:mm。 <ul style="list-style-type: none"> • 维护时间窗开始和结束时间必须为指定的时间段。 • 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时，结束时间为02:00。 • 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。
enable_publicip	否	Boolean	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 不开启
publicip_id	否	String	RabbitMQ实例绑定的弹性IP地址的ID。 如果开启了公网访问功能（即enable_publicip为true），该字段为必选。

参数	是否必选	参数类型	描述
ssl_enable	否	Boolean	是否打开SSL加密访问。 <ul style="list-style-type: none"> • true: 打开SSL加密访问。 • false: 不打开SSL加密访问。
storage_spec_code	是	String	存储IO规格。 如何选择磁盘类型请参考 磁盘类型及性能介绍 。 取值范围: <ul style="list-style-type: none"> • dms.physical.storage.high.v2 • dms.physical.storage.ultra.v2 • dms.physical.storage.high.dss.v2(专属云) • dms.physical.storage.ultra.dss.v2(专属云)
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。
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 TagEntity

参数	是否必选	参数类型	描述
key	否	String	标签键。 <ul style="list-style-type: none"> • 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	是否必选	参数类型	描述
value	否	String	<p>标签值。</p> <ul style="list-style-type: none"> 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 5-4 BssParam

参数	是否必选	参数类型	描述
is_auto_renew	否	Boolean	<p>是否自动续订。</p> <p>取值范围：</p> <ul style="list-style-type: none"> true: 自动续订。 false: 不自动续订。 <p>默认不自动续订。</p>
charging_mode	否	String	<p>计费模式。</p> <p>功能说明：付费方式。</p> <p>取值范围：</p> <ul style="list-style-type: none"> prePaid: 预付费，即包年包月； postPaid: 后付费，即按需付费； <p>默认为postPaid。</p>
is_auto_pay	否	Boolean	<p>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。</p> <p>取值范围：</p> <ul style="list-style-type: none"> true: 是（自动支付） false: 否（需要客户手动支付） <p>默认为手动支付。</p>
period_type	否	String	<p>订购周期类型。</p> <p>取值范围：</p> <ul style="list-style-type: none"> month: 月 year: 年 <p>chargingMode为prePaid时生效且为必选值。</p>

参数	是否必选	参数类型	描述
period_num	否	Integer	订购周期数。 取值范围： <ul style="list-style-type: none"> 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 5-5 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RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

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

```
{
  "name": "rabbitmq-demo",
  "description": "",
  "engine": "RabbitMQ",
  "engine_version": "3.8.35",
  "storage_space": 100,
  "access_user": "*****",
  "password": "*****",
  "vpc_id": "1e93f86e-13af-46c8-97d6-d40fa62b76c2",
  "security_group_id": "0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
  "subnet_id": "b5fa806c-35e7-4299-b659-b39398dd4718",
  "available_zones": [ "d573142f24894ef3bd3664de068b44b0" ],
  "product_id": "c6.2u4g.single",
  "ssl_enable": false,
  "enable_publicip": false,
  "publicip_id": "",
  "storage_spec_code": "dms.physical.storage.high.v2"
}
```

- 创建一个包年包月的RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

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

```
{
  "name": "rabbitmq-demo",
  "description": "",
  "engine": "RabbitMQ",
}
```

```
"engine_version" : "3.8.35",
"storage_space" : 100,
"access_user" : "*****",
"password" : "*****",
"vpc_id" : "1e93f86e-13af-46c8-97d6-d40fa62b76c2",
"security_group_id" : "0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
"subnet_id" : "b5fa806c-35e7-4299-b659-b39398dd4718",
"available_zones" : [ "d573142f24894ef3bd3664de068b44b0" ],
"product_id" : "c6.2u4g.single",
"ssl_enable" : false,
"enable_publicip" : false,
"publicip_id" : "",
"storage_spec_code" : "dms.physical.storage.high.v2",
"bss_param" : {
  "charging_mode" : "prePaid",
  "period_type" : "month",
  "period_num" : 1,
  "is_auto_pay" : true
}
}
```

- 创建一个RabbitMQ实例，版本为AMQP-0-9-1，规格为amqp.b1.large.1，100GB的存储空间。

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

```
{
  "name" : "rabbitmq-aor-demo",
  "description" : "",
  "engine" : "RabbitMQ",
  "engine_version" : "AMQP-0-9-1",
  "storage_space" : 100,
  "vpc_id" : "05590544-f553-4158-be38-c791589ad303",
  "security_group_id" : "030f635d-b407-4ffb-b530-6b4eaf8edc03",
  "subnet_id" : "89c1cb26-787b-4d66-a6e4-1bd887f19183",
  "available_zones" : [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
  "product_id" : "amqp.b1.large.1",
  "broker_num" : 1,
  "ssl_enable" : false,
  "enable_publicip" : false,
  "storage_spec_code" : "dms.physical.storage.high.v2",
  "enable_acl" : true,
  "enterprise_project_id" : 0
}
```

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RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class CreatePostPaidInstanceByEngineSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreatePostPaidInstanceByEngineRequest request = new
        CreatePostPaidInstanceByEngineRequest();

        request.withEngine(CreatePostPaidInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
        CreateInstanceReq body = new CreateInstanceReq();
        List<String> listbodyAvailableZones = new ArrayList<>();
        listbodyAvailableZones.add("d573142f24894ef3bd3664de068b44b0");

        body.withStorageSpecCode(CreateInstanceReq.StorageSpecCodeEnum.fromValue("dms.physical.storage
        e.high.v2"));
        body.withSslEnable(false);
        body.withPublicId("");
        body.withEnablePublicip(false);
        body.withProductId("c6.2u4g.single");
        body.withAvailableZones(listbodyAvailableZones);
        body.withSubnetId("b5fa806c-35e7-4299-b659-b39398dd4718");
        body.withSecurityGroupId("0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8");
        body.withVpId("1e93f86e-13af-46c8-97d6-d40fa62b76c2");
        body.withPassword("*****");
        body.withAccessUser("*****");
        body.withStorageSpace(100);
        body.withEngineVersion("3.8.35");
        body.withEngine(CreateInstanceReq.EngineEnum.fromValue("RabbitMQ"));
        body.withDescription("");
        body.withName("rabbitmq-demo");
        request.withBody(body);
        try {
            CreatePostPaidInstanceByEngineResponse response =
            client.createPostPaidInstanceByEngine(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.getStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
}
}

```

- 创建一个包年包月的RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class CreatePostPaidInstanceByEngineSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreatePostPaidInstanceByEngineRequest request = new
        CreatePostPaidInstanceByEngineRequest();

        request.withEngine(CreatePostPaidInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
        CreateInstanceReq body = new CreateInstanceReq();
        BssParam bssParambody = new BssParam();
        bssParambody.withChargingMode(BssParam.ChargingModeEnum.fromValue("prePaid"))
            .withIsAutoPay(true)
            .withPeriodType(BssParam.PeriodTypeEnum.fromValue("month"))
            .withPeriodNum(1);
        List<String> listbodyAvailableZones = new ArrayList<>();
        listbodyAvailableZones.add("d573142f24894ef3bd3664de068b44b0");
        body.withBssParam(bssParambody);

        body.withStorageSpecCode(CreateInstanceReq.StorageSpecCodeEnum.fromValue("dms.physical.storage
        e.high.v2"));
        body.withSslEnable(false);
        body.withPublicIpId("");
        body.withEnablePublicIp(false);
        body.withProductId("c6.2u4g.single");
        body.withAvailableZones(listbodyAvailableZones);
        body.withSubnetId("b5fa806c-35e7-4299-b659-b39398dd4718");
        body.withSecurityGroupId("0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8");
    }
}

```

```

body.withVpcId("1e93f86e-13af-46c8-97d6-d40fa62b76c2");
body.withPassword("*****");
body.withAccessUser("*****");
body.withStorageSpace(100);
body.withEngineVersion("3.8.35");
body.withEngine(CreateInstanceReq.EngineEnum.fromValue("RabbitMQ"));
body.withDescription("");
body.withName("rabbitmq-demo");
request.withBody(body);
try {
    CreatePostPaidInstanceByEngineResponse response =
client.createPostPaidInstanceByEngine(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RabbitMQ实例，版本为AMQP-0-9-1，规格为amqp.b1.large.1，100GB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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class CreatePostPaidInstanceByEngineSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreatePostPaidInstanceByEngineRequest request = new
        CreatePostPaidInstanceByEngineRequest();

        request.withEngine(CreatePostPaidInstanceByEngineRequest.EngineEnum.fromValue("{engine}"));
    }
}

```

```

CreateInstanceReq body = new CreateInstanceReq();
List<String> listbodyAvailableZones = new ArrayList<>();
listbodyAvailableZones.add("9f1c5806706d4c1fb0eb72f0a9b18c77");
body.withEnterpriseProjectId("0");

body.withStorageSpecCode(CreateInstanceReq.StorageSpecCodeEnum.fromValue("dms.physical.storage.high.v2"));
body.withSslEnable(false);
body.withEnablePublicip(false);
body.withBrokerNum(CreateInstanceReq.BrokerNumEnum.NUMBER_1);
body.withProductId("amqp.b1.large.1");
body.withAvailableZones(listbodyAvailableZones);
body.withSubnetId("89c1cb26-787b-4d66-a6e4-1bd887f19183");
body.withSecurityGroupId("030f635d-b407-4ffb-b530-6b4eaf8edc03");
body.withVpcId("05590544-f553-4158-be38-c791589ad303");
body.withStorageSpace(100);
body.withEnableAcl(true);
body.withEngineVersion("AMQP-0-9-1");
body.withEngine(CreateInstanceReq.EngineEnum.fromValue("RabbitMQ"));
body.withDescription("");
body.withName("rabbitmq-aor-demo");
request.withBody(body);
try {
    CreatePostPaidInstanceByEngineResponse response =
client.createPostPaidInstanceByEngine(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RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \

```

```

.build()

try:
    request = CreatePostPaidInstanceByEngineRequest()
    request.engine = "{engine}"
    listAvailableZonesbody = [
        "d573142f24894ef3bd3664de068b44b0"
    ]
    request.body = CreateInstanceReq(
        storage_spec_code="dms.physical.storage.high.v2",
        ssl_enable=False,
        publicip_id="",
        enable_publicip=False,
        product_id="c6.2u4g.single",
        available_zones=listAvailableZonesbody,
        subnet_id="b5fa806c-35e7-4299-b659-b39398dd4718",
        security_group_id="0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
        vpc_id="1e93f86e-13af-46c8-97d6-d40fa62b76c2",
        password="*****",
        access_user="*****",
        storage_space=100,
        engine_version="3.8.35",
        engine="RabbitMQ",
        description="",
        name="rabbitmq-demo"
    )
    response = client.create_post_paid_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RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreatePostPaidInstanceByEngineRequest()
        request.engine = "{engine}"
        bssParambody = BssParam(
            charging_mode="prePaid",
            is_auto_pay=True,
            period_type="month",

```

```

        period_num=1
    )
    listAvailableZonesbody = [
        "d573142f24894ef3bd3664de068b44b0"
    ]
    request.body = CreateInstanceReq(
        bss_param=bssParambody,
        storage_spec_code="dms.physical.storage.high.v2",
        ssl_enable=False,
        publicip_id="",
        enable_publicip=False,
        product_id="c6.2u4g.single",
        available_zones=listAvailableZonesbody,
        subnet_id="b5fa806c-35e7-4299-b659-b39398dd4718",
        security_group_id="0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
        vpc_id="1e93f86e-13af-46c8-97d6-d40fa62b76c2",
        password="*****",
        access_user="*****",
        storage_space=100,
        engine_version="3.8.35",
        engine="RabbitMQ",
        description="",
        name="rabbitmq-demo"
    )
    response = client.create_post_paid_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RabbitMQ实例，版本为AMQP-0-9-1，规格为amqp.b1.large.1，100GB的存储空间。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudskrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudskrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreatePostPaidInstanceByEngineRequest()
        request.engine = "{engine}"
        listAvailableZonesbody = [
            "9f1c5806706d4c1fb0eb72f0a9b18c77"
        ]
        request.body = CreateInstanceReq(
            enterprise_project_id="0",
            storage_spec_code="dms.physical.storage.high.v2",

```



```

        ssl_enable=False,
        enable_publicip=False,
        broker_num=1,
        product_id="amqp.b1.large.1",
        available_zones=listAvailableZonesbody,
        subnet_id="89c1cb26-787b-4d66-a6e4-1bd887f19183",
        security_group_id="030f635d-b407-4ffb-b530-6b4eaf8edc03",
        vpc_id="05590544-f553-4158-be38-c791589ad303",
        storage_space=100,
        enable_acl=True,
        engine_version="AMQP-0-9-1",
        engine="RabbitMQ",
        description="",
        name="rabbitmq-aor-demo"
    )
    response = client.create_post_paid_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RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

```

package main

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

    request := &model.CreatePostPaidInstanceByEngineRequest{}
    request.Engine = model.GetCreatePostPaidInstanceByEngineRequestEngineEnum().ENGINE
    var listAvailableZonesbody = []string{
        "d573142f24894ef3bd3664de068b44b0",
    }
    }
    sslEnableCreateInstanceReq:= false
    publicIpCreateInstanceReq:= ""
    enablePublicIpCreateInstanceReq:= false

```

```

passwordCreateInstanceReq:= "*****"
accessUserCreateInstanceReq:= "*****"
descriptionCreateInstanceReq:= ""
request.Body = &model.CreateInstanceReq{
    StorageSpecCode:
model.GetCreateInstanceReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_HIGH,
    SslEnable: &sslEnableCreateInstanceReq,
    PublicIpId: &publicIpIdCreateInstanceReq,
    EnablePublicIp: &enablePublicIpCreateInstanceReq,
    ProductId: "c6.2u4g.single",
    AvailableZones: listAvailableZonesbody,
    SubnetId: "b5fa806c-35e7-4299-b659-b39398dd4718",
    SecurityGroupId: "0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
    VpcId: "1e93f86e-13af-46c8-97d6-d40fa62b76c2",
    Password: &passwordCreateInstanceReq,
    AccessUser: &accessUserCreateInstanceReq,
    StorageSpace: int32(100),
    EngineVersion: "3.8.35",
    Engine: model.GetCreateInstanceReqEngineEnum().RABBIT_MQ,
    Description: &descriptionCreateInstanceReq,
    Name: "rabbitmq-demo",
}
response, err := client.CreatePostPaidInstanceByEngine(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
}

```

- 创建一个包年包月的RabbitMQ实例，版本为3.8.35，规格为2U4G*1，100GB的存储空间。

```

package main

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

    request := &model.CreatePostPaidInstanceByEngineRequest{
        request.Engine = model.GetCreatePostPaidInstanceByEngineRequestEngineEnum().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,
}
var listAvailableZonesbody = []string{
    "d573142f24894ef3bd3664de068b44b0",
}
sslEnableCreateInstanceReq:= false
publicIpIdCreateInstanceReq:= ""
enablePublicIpCreateInstanceReq:= false
passwordCreateInstanceReq:= "*****"
accessUserCreateInstanceReq:= "*****"
descriptionCreateInstanceReq:= ""
request.Body = &model.CreateInstanceReq{
    BssParam: bssParambody,
    StorageSpecCode:
model.GetCreateInstanceReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_HIGH,
    SslEnable: &sslEnableCreateInstanceReq,
    PublicIpId: &publicIpIdCreateInstanceReq,
    EnablePublicIp: &enablePublicIpCreateInstanceReq,
    ProductId: "c6.2u4g.single",
    AvailableZones: listAvailableZonesbody,
    SubnetId: "b5fa806c-35e7-4299-b659-b39398dd4718",
    SecurityGroupId: "0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
    VpId: "1e93f86e-13af-46c8-97d6-d40fa62b76c2",
    Password: &passwordCreateInstanceReq,
    AccessUser: &accessUserCreateInstanceReq,
    StorageSpace: int32(100),
    EngineVersion: "3.8.35",
    Engine: model.GetCreateInstanceReqEngineEnum().RABBIT_MQ,
    Description: &descriptionCreateInstanceReq,
    Name: "rabbitmq-demo",
}
response, err := client.CreatePostPaidInstanceByEngine(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
}

```

- 创建一个RabbitMQ实例，版本为AMQP-0-9-1，规格为amqp.b1.large.1，100GB的存储空间。

```

package main

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

request := &model.CreatePostPaidInstanceByEngineRequest{}
request.Engine = model.GetCreatePostPaidInstanceByEngineRequestEngineEnum().ENGINE
var listAvailableZonesbody = []string{
    "9f1c5806706d4c1fb0eb72f0a9b18c77",
}
enterpriseProjectIdCreateInstanceReq:= "0"
sslEnableCreateInstanceReq:= false
enablePublicipCreateInstanceReq:= false
brokerNumCreateInstanceReq:= model.GetCreateInstanceReqBrokerNumEnum().E_1
enableAclCreateInstanceReq:= true
descriptionCreateInstanceReq:= ""
request.Body = &model.CreateInstanceReq{
    EnterpriseProjectId: &enterpriseProjectIdCreateInstanceReq,
    StorageSpecCode:
model.GetCreateInstanceReqStorageSpecCodeEnum().DMS_PHYSICAL_STORAGE_HIGH,
    SslEnable: &sslEnableCreateInstanceReq,
    EnablePublicip: &enablePublicipCreateInstanceReq,
    BrokerNum: &brokerNumCreateInstanceReq,
    ProductId: "amqp.b1.large.1",
    AvailableZones: listAvailableZonesbody,
    SubnetId: "89c1cb26-787b-4d66-a6e4-1bd887f19183",
    SecurityGroupId: "030f635d-b407-4ffb-b530-6b4eaf8edc03",
    VpcId: "05590544-f553-4158-be38-c791589ad303",
    StorageSpace: int32(100),
    EnableAcl: &enableAclCreateInstanceReq,
    EngineVersion: "AMQP-0-9-1",
    Engine: model.GetCreateInstanceReqEngineEnum().RABBIT_MQ,
    Description: &descriptionCreateInstanceReq,
    Name: "rabbitmq-aor-demo",
}
response, err := client.CreatePostPaidInstanceByEngine(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 5-6 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 5-7 Query 参数

参数	是否必选	参数类型	描述
engine	是	String	引擎类型：rabbitmq。
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_failed	否	String	是否返回创建失败的实例数。当参数值为“true”时，返回创建失败的实例数。参数值为“false”或者其他值，不返回创建失败的实例数。
exact_match_name	否	String	是否按照实例名称进行精确匹配查询。 默认为“false”，表示模糊匹配实例名称查询。若参数值为“true”表示按照实例名称进行精确匹配查询。
enterprise_project_id	否	String	企业项目ID。
offset	否	String	偏移量，表示从此偏移量开始查询，offset大于等于0。

参数	是否必选	参数类型	描述
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 5-8 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	实例个数。

表 5-9 ShowInstanceResp

参数	参数类型	描述
access_user	String	认证用户名，只能由英文字母、数字、中划线组成，长度为4~64的字符。
broker_num	Integer	代理个数。
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。 <ul style="list-style-type: none"> RabbitMQ实例单机返回vm规格。 RabbitMQ实例集群返回vm规格和节点数。
storage_space	Integer	消息存储空间，单位：GB。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。

参数	参数类型	描述
dns_enable	Boolean	实例是否开启域名访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
connect_address	String	实例内网连接IP地址。
connect_domain_name	String	实例内网连接域名。
public_connect_address	String	实例公网连接IP地址。
public_connect_domain_name	String	实例公网连接域名。
port	Integer	实例连接端口。
status	String	实例的状态。
description	String	实例描述。
instance_id	String	实例ID。

参数	参数类型	描述
resource_spec_code	String	<p>资源规格标识。</p> <ul style="list-style-type: none"> dms.instance.rabbitmq.single.c3.2u4g: RabbitMQ单机, vm规格2u4g dms.instance.rabbitmq.single.c3.4u8g: RabbitMQ单机, vm规格4u8g dms.instance.rabbitmq.single.c3.8u16g: RabbitMQ单机, vm规格8u16g dms.instance.rabbitmq.single.c3.16u32g: RabbitMQ单机, vm规格16u32g dms.instance.rabbitmq.cluster.c3.4u8g.3: RabbitMQ集群, vm规格4u8g, 3个节点 dms.instance.rabbitmq.cluster.c3.4u8g.5: RabbitMQ集群, vm规格4u8g, 5个节点 dms.instance.rabbitmq.cluster.c3.4u8g.7: RabbitMQ集群, vm规格4u8g, 7个节点 dms.instance.rabbitmq.cluster.c3.8u16g.3: RabbitMQ集群, vm规格8u16g, 3个节点 dms.instance.rabbitmq.cluster.c3.8u16g.5: RabbitMQ集群, vm规格8u16g, 5个节点 dms.instance.rabbitmq.cluster.c3.8u16g.7: RabbitMQ集群, vm规格8u16g, 7个节点 dms.instance.rabbitmq.cluster.c3.16u32g.3: RabbitMQ集群, vm规格16u32g, 3个节点 dms.instance.rabbitmq.cluster.c3.16u32g.5: RabbitMQ集群, vm规格16u32g, 5个节点 dms.instance.rabbitmq.cluster.c3.16u32g.7: RabbitMQ集群, vm规格16u32g, 7个节点
charging_mode	Integer	<p>付费模式, 1表示按需计费, 0表示包年/包月计费。</p>
vpc_id	String	<p>VPC ID。</p>
vpc_name	String	<p>VPC的名称。</p>

参数	参数类型	描述
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。
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	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
publicip_address	String	RabbitMQ实例绑定的弹性IP地址。如果未开启公网访问功能，该字段值为 null。
publicip_id	String	RabbitMQ实例绑定的弹性IP地址的ID。如果未开启公网访问功能，该字段值为 null。
management_connect_address	String	RabbitMQ实例的管理地址。
management_connect_domain_name	String	RabbitMQ实例的管理域名。
public_management_connect_addresses	String	RabbitMQ实例的公网管理地址。
public_management_connect_domain_name	String	RabbitMQ实例的公网管理域名。
ssl_enable	Boolean	是否开启安全认证。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
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。 <ul style="list-style-type: none"> • 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。
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, 返回“可用区名称”。
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。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
tags	Array of TagEntity objects	标签列表。

表 5-10 TagEntity

参数	参数类型	描述
key	String	标签键。 <ul style="list-style-type: none"> 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。 <ul style="list-style-type: none"> 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": "api-explorer",
    "engine": "rabbitmq",
    "port": 5672,
    "status": "RUNNING",
    "type": "single",
    "specification": "2vCPUs 4GB",
    "engine_version": "3.8.35",
    "connect_address": "192.168.0.74",
    "instance_id": "de873040-d661-4770-aa96-9329c71d7c8a",
    "resource_spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
    "charging_mode": 1,
    "vpc_id": "40a6501e-85ca-4449-a0db-b8bc7f0cec28",
    "vpc_name": "vpc-a400",
    "created_at": "1590047080687",
    "product_id": "00300-30109-0--0",
    "security_group_id": "bfd68e26-f8ef-4a91-a373-0a8f5c198601",
    "security_group_name": "Sys-default",
    "subnet_id": "a7f9a564-30dd-4059-8124-364ca6554578",
    "available_zones": [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
    "available_zone_names": [ "AZ1" ],
    "user_id": "3df5acbc24a54fadb62a043c9000a307",
    "user_name": "*****",
  } ]
}
```

```

"maintain_begin" : "22:00:00",
"maintain_end" : "02:00:00",
"storage_space" : 88,
"total_storage_space" : 100,
"used_storage_space" : 4,
"enable_publicip" : false,
"ssl_enable" : false,
"management_connect_address" : "http://192.168.0.74:15672",
"storage_resource_id" : "52be287d-1d6a-4d30-937e-185b3f176fc4",
"storage_spec_code" : "dms.physical.storage.normal",
"enterprise_project_id" : "0",
"tags" : [{
  "key" : "key1",
  "value" : "value1"
}, {
  "key" : "key2",
  "value" : "value2"
}],
"is_logical_volume" : true,
"extend_times" : 0,
"ipv6_enable" : false,
"ipv6_connect_addresses" : []
}],
"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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListInstancesDetailsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ListInstancesDetailsRequest request = new ListInstancesDetailsRequest();
        try {
            ListInstancesDetailsResponse response = client.listInstancesDetails(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListInstancesDetailsRequest()
        response = client.list_instances_details(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

```

Go

```

package main

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

request := &model.ListInstancesDetailsRequest{}
response, err := client.ListInstancesDetails(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 5-11 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 5-12 响应 Body 参数

参数	参数类型	描述
access_user	String	认证用户名, 只能由英文字母、数字、中划线组成, 长度为4~64的字符。
broker_num	Integer	代理个数。
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。 <ul style="list-style-type: none"> RabbitMQ实例单机返回vm规格。 RabbitMQ实例集群返回vm规格和节点数。
storage_space	Integer	消息存储空间, 单位: GB。
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。 <ul style="list-style-type: none"> true: 开启 false: 未开启
connect_address	String	实例内网连接IP地址。
connect_domain_name	String	实例内网连接域名。
public_connect_address	String	实例公网连接IP地址。

参数	参数类型	描述
public_connect_domain_name	String	实例公网连接域名。
port	Integer	实例连接端口。
status	String	实例的状态。
description	String	实例描述。
instance_id	String	实例ID。

参数	参数类型	描述
resource_spec_code	String	<p>资源规格标识。</p> <ul style="list-style-type: none"> dms.instance.rabbitmq.single.c3.2u4g: RabbitMQ单机, vm规格2u4g dms.instance.rabbitmq.single.c3.4u8g: RabbitMQ单机, vm规格4u8g dms.instance.rabbitmq.single.c3.8u16g: RabbitMQ单机, vm规格8u16g dms.instance.rabbitmq.single.c3.16u32g: RabbitMQ单机, vm规格16u32g dms.instance.rabbitmq.cluster.c3.4u8g.3: RabbitMQ集群, vm规格4u8g, 3个节点 dms.instance.rabbitmq.cluster.c3.4u8g.5: RabbitMQ集群, vm规格4u8g, 5个节点 dms.instance.rabbitmq.cluster.c3.4u8g.7: RabbitMQ集群, vm规格4u8g, 7个节点 dms.instance.rabbitmq.cluster.c3.8u16g.3: RabbitMQ集群, vm规格8u16g, 3个节点 dms.instance.rabbitmq.cluster.c3.8u16g.5: RabbitMQ集群, vm规格8u16g, 5个节点 dms.instance.rabbitmq.cluster.c3.8u16g.7: RabbitMQ集群, vm规格8u16g, 7个节点 dms.instance.rabbitmq.cluster.c3.16u32g.3: RabbitMQ集群, vm规格16u32g, 3个节点 dms.instance.rabbitmq.cluster.c3.16u32g.5: RabbitMQ集群, vm规格16u32g, 5个节点 dms.instance.rabbitmq.cluster.c3.16u32g.7: RabbitMQ集群, vm规格16u32g, 7个节点
charging_mode	Integer	<p>付费模式, 1表示按需计费, 0表示包年/包月计费。</p>
vpc_id	String	<p>VPC ID。</p>
vpc_name	String	<p>VPC的名称。</p>

参数	参数类型	描述
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。
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	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
publicip_address	String	RabbitMQ实例绑定的弹性IP地址。如果未开启公网访问功能，该字段值为 null。
publicip_id	String	RabbitMQ实例绑定的弹性IP地址的ID。如果未开启公网访问功能，该字段值为 null。
management_connect_address	String	RabbitMQ实例的管理地址。
management_connect_domain_name	String	RabbitMQ实例的管理域名。
public_management_connect_addresses	String	RabbitMQ实例的公网管理地址。
public_management_connect_domain_name	String	RabbitMQ实例的公网管理域名。
ssl_enable	Boolean	是否开启安全认证。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
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。 <ul style="list-style-type: none"> • 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。
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, 返回“可用区名称”。
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。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
tags	Array of TagEntity objects	标签列表。

表 5-13 TagEntity

参数	参数类型	描述
key	String	标签键。 <ul style="list-style-type: none"> 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。 <ul style="list-style-type: none"> 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

查询指定实例的详细信息。

```
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": "api-explorer",
  "engine": "rabbitmq",
  "port": 5672,
  "status": "RUNNING",
  "type": "single",
  "specification": "2vCPUs 4GB",
  "engine_version": "3.8.35",
  "connect_address": "192.168.0.74",
  "instance_id": "de873040-d661-4770-aa96-9329c71d7c8a",
  "resource_spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
  "charging_mode": 1,
  "vpc_id": "40a6501e-85ca-4449-a0db-b8bc7f0cec28",
  "vpc_name": "vpc-a400",
  "created_at": "1590047080687",
  "product_id": "00300-30109-0--0",
  "security_group_id": "bfd68e26-f8ef-4a91-a373-0a8f5c198601",
  "security_group_name": "Sys-default",
  "subnet_id": "a7f9a564-30dd-4059-8124-364ca6554578",
  "available_zones": [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
  "available_zone_names": [ "AZ1" ],
  "user_id": "3df5acbc24a54fadb62a043c9000a307",
  "user_name": "paas_dms_01",
  "maintain_begin": "22:00:00",
```

```
"maintain_end" : "02:00:00",
"storage_space" : 88,
"total_storage_space" : 100,
"used_storage_space" : 4,
"enable_publicip" : false,
"ssl_enable" : false,
"management_connect_address" : "http://192.168.0.74:15672",
"storage_resource_id" : "52be287d-1d6a-4d30-937e-185b3f176fc4",
"storage_spec_code" : "dms.physical.storage.normal",
"enterprise_project_id" : "0",
"tags" : [ {
  "key" : "key1",
  "value" : "value1"
}, {
  "key" : "key2",
  "value" : "value2"
} ],
"is_logical_volume" : true,
"extend_times" : 0,
"ipv6_enable" : false,
"ipv6_connect_addresses" : [ ],
"broker_num" : 1,
"access_user" : "root_01"
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        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 5-14 路径参数

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

请求参数

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
"github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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	删除实例成功。

错误码

请参见[错误码](#)。

5.1.5 修改实例信息

功能介绍

修改实例的名称和描述信息。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-15 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 5-16 请求 Body 参数

参数	是否必选	参数类型	描述
name	否	String	实例名称。 由英文字符开头, 只能由英文字母、数字、中划线组成, 长度为4~64的字符。
description	否	String	实例的描述信息。 长度不超过1024的字符串。 说明 \与"在json报文中属于特殊字符, 如果参数值中需要显示\或者"字符, 请在字符前增加转义字符\, 比如\或者"。
maintain_begin	否	String	维护时间窗开始时间, 格式为HH:mm:ss。 <ul style="list-style-type: none"> 维护时间窗开始和结束时间必须为指定的时间段。 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。

参数	是否必选	参数类型	描述
maintain_end	否	String	维护时间窗结束时间，格式为 HH:mm:ss。 <ul style="list-style-type: none"> 维护时间窗开始和结束时间必须为指定的时间段。 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	否	String	安全组 ID。 获取方法如下：登录虚拟私有云服务的控制台界面，在安全组的详情页面查找安全组 ID。
enable_publicip	否	Boolean	RabbitMQ 实例是否开启公网访问功能。 <ul style="list-style-type: none"> true: 开启 false: 不开启
publicip_id	否	String	RabbitMQ 实例绑定的弹性 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。 获取方法：登录弹性公网 IP 和带宽的控制台界面，在弹性公网 IP 的详情页面查的基本信息栏找 ID。
enterprise_project_id	否	String	企业项目。
enable_acl	否	Boolean	ACL 访问控制（仅 AMQP 版本支持此参数）。

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": "rabbitmq-01",  
"description": "instance 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": "rabbitmq-01",  
  "description": "instance description",  
  "maintain_begin": "02:00:00",  
  "maintain_end": "06:00:00"  
}
```

- 开启公网访问。

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

```
{  
  "enable_publicip": true,  
  "publicip_id": "32685c2b-xxxx-xxxx-86c6-a1902359xxxx"  
}
```

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.rabbitmq.v2.region.RabbitMQRegion;  
import com.huaweicloud.sdk.rabbitmq.v2.*;  
import com.huaweicloud.sdk.rabbitmq.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);  
  
        RabbitMQClient client = RabbitMQClient.newBuilder()  
            .withCredential(auth)  
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
```

```

        .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withDescription("instance description");
        body.withName("rabbitmq-01");
        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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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("rabbitmq-01");
        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());
    }
}
}
}

```

- 开启公网访问。

```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        UpdateInstanceRequest request = new UpdateInstanceRequest();
        request.withInstanceId("{instance_id}");
        UpdateInstanceReq body = new UpdateInstanceReq();
        body.withPublicId("32685c2b-xxxx-xxxx-86c6-a1902359xxxx");
        body.withEnablePublicip(true);
        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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
            description="instance description",
            name="rabbitmq-01"
        )
        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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.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="rabbitmq-01"
    )
    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)

```

- 开启公网访问。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = UpdateInstanceRequest()
        request.instance_id = "{instance_id}"
        request.body = UpdateInstanceReq(
            publicip_id="32685c2b-xxxx-xxxx-86c6-a1902359xxxx",
            enable_publicip=True
        )
        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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.UpdateInstanceRequest{
        request.InstanceId = "{instance_id}"
        descriptionUpdateInstanceReq:= "instance description"
        nameUpdateInstanceReq:= "rabbitmq-01"
        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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        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:= "rabbitmq-01"
    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)
}
}

```

- 开启公网访问。

```

package main

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

    request := &model.UpdateInstanceRequest{
        request.InstanceId = "{instance_id}"
        publicIdUpdateInstanceReq:= "32685c2b-xxxx-xxxx-86c6-a1902359xxxx"
    }
}

```

```
enablePublicipUpdateInstanceReq:= true
request.Body = &model.UpdateInstanceReq{
    PublicipId: &publicipIdUpdateInstanceReq,
    EnablePublicip: &enablePublicipUpdateInstanceReq,
}
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	修改实例成功。

错误码

请参见[错误码](#)。

5.1.6 批量删除实例

功能介绍

批量删除实例。

实例删除后，实例中原有的数据将被删除，且没有备份，请谨慎操作。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/{project_id}/instances/action

表 5-17 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 。

请求参数

表 5-18 请求 Body 参数

参数	是否必选	参数类型	描述
instances	否	Array of strings	实例的ID列表。
action	是	String	对实例的操作: delete
all_failure	否	String	是否批量删除创建失败的实例。 当参数值为“rabbitmq”时，删除租户所有创建失败的实例，此时请求参数instances可为空。

响应参数

状态码: 200

表 5-19 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 5-20 results

参数	参数类型	描述
result	String	操作结果: <ul style="list-style-type: none"> • success: 操作成功 • failed: 操作失败
instance	String	实例ID。

请求示例

- 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": "rabbitmq"
}
```

响应示例

状态码：200

批量删除实例成功。

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

SDK 代码示例

SDK代码示例如下。

Java

- 批量删除实例。

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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("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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        BatchRestartOrDeleteInstancesRequest request = new BatchRestartOrDeleteInstancesRequest();
        BatchRestartOrDeleteInstanceReq body = new BatchRestartOrDeleteInstanceReq();
        body.withAllFailure(BatchRestartOrDeleteInstanceReq.AllFailureEnum.fromValue("rabbitmq"));
        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

- 批量删除实例。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
    .build()

try:
    request = BatchRestartOrDeleteInstancesRequest()
    request.body = BatchRestartOrDeleteInstanceReq(
        all_failure="rabbitmq",
        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

- 批量删除实例。

```

package main

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

    request := &model.BatchRestartOrDeleteInstancesRequest{}
    allFailureBatchRestartOrDeleteInstanceReq:=
    model.GetBatchRestartOrDeleteInstanceReqAllFailureEnum().RABBITMQ
    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	批量删除实例成功。
204	删除所有创建失败的RabbitMQ实例成功。

错误码

请参见[错误码](#)。

5.1.7 开启 RabbitMQ 实例域名访问能力

功能介绍

开启RabbitMQ实例域名访问功能后，客户端可以通过域名连接RabbitMQ实例。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/{project_id}/rabbitmq/instances/{instance_id}/dns

表 5-21 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，从 查询所有实例列表 获取实例ID。

请求参数

无

响应参数

状态码：400

表 5-22 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

开启RabbitMQ实例域名访问能力。

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

响应示例

状态码：400

开启域名任务提交失败。

```
{
  "error_code" : "DMS.111501035",
  "error_msg" : "dns already enabled"
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class EnableDnsSolution {

    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);

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

```

try {
    EnableDnsResponse response = client.enableDns(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = EnableDnsRequest()
        request.instance_id = "{instance_id}"
        response = client.enable_dns(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        WithRegion(region.ValueOf("<YOUR REGION>")).
        WithCredential(auth).
        Build())

request := &model.EnableDnsRequest{}
request.InstanceId = "{instance_id}"
response, err := client.EnableDns(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	开启域名任务提交成功。
400	开启域名任务提交失败。

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。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-23 路径参数

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

请求参数

表 5-24 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： <ul style="list-style-type: none">● 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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        WithRegion(region.ValueOf("<YOUR REGION>")).
        WithCredential(auth).
        Build())

request := &model.ResetPasswordRequest{}
request.InstanceId = "{instance_id}"
newPasswordResetPasswordReq:= "*****"
request.Body = &model.ResetPasswordReq{
    NewPassword: &newPasswordResetPasswordReq,
}
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 查询插件列表

功能介绍

查询插件列表。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-25 路径参数

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

参数	是否必选	参数类型	描述
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码：200

表 5-26 响应 Body 参数

参数	参数类型	描述
plugins	Array of PluginEntity objects	插件信息列表。

表 5-27 PluginEntity

参数	参数类型	描述
running	Boolean	是否运行。
enable	Boolean	是否启用。
name	String	插件名称。
version	String	插件版本。

请求示例

查询插件列表。

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

响应示例

状态码：200

查询插件列表成功。

```
{  
  "plugins": [{  
    "running": true,  
    "enable": true,  
    "name": "rabbitmq_shovel",  
    "version": "3.8.35"  
  }, {  
    "running": true,
```

```

    "enable" : true,
    "name" : "rabbitmq_consistent_hash_exchange",
    "version" : "3.8.35"
  }, {
    "running" : false,
    "enable" : false,
    "name" : "rabbitmq_federation",
    "version" : "3.8.35"
  }
}

```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListPluginsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ListPluginsRequest request = new ListPluginsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ListPluginsResponse response = client.listPlugins(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListPluginsRequest()
        request.instance_id = "{instance_id}"
        response = client.list_plugins(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
```

```
WithCredential(auth).  
Build())  
  
request := &model.ListPluginsRequest{}  
request.InstanceId = "{instance_id}"  
response, err := client.ListPlugins(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	查询插件列表成功。

错误码

请参见[错误码](#)。

5.2.3 开启或关闭插件

功能介绍

开启或关闭插件。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-28 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 5-29 请求 Body 参数

参数	是否必选	参数类型	描述
enable	否	Boolean	是否开启该插件。
plugins	否	String	插件列表，多个插件中间用“,” 隔开。

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	后台任务ID。

请求示例

开启rabbitmq_federation和rabbitmq_shovel插件。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/rabbitmq/plugins
{
  "enable": true,
  "plugins": "rabbitmq_federation,rabbitmq_shovel"
}
```

响应示例

状态码：200

开启或关闭插件成功。

```
{
  "job_id": "8abfa7b27437db8f01744ea8ad4f245e"
}
```

SDK 代码示例

SDK代码示例如下。

Java

开启rabbitmq_federation和rabbitmq_shovel插件。

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class UpdatePluginsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        UpdatePluginsRequest request = new UpdatePluginsRequest();
        request.withInstanceId("{instance_id}");
        UpdatePluginsReq body = new UpdatePluginsReq();
        body.withPlugins("rabbitmq_federation,rabbitmq_shovel");
        body.withEnable(true);
        request.withBody(body);
        try {
            UpdatePluginsResponse response = client.updatePlugins(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

开启rabbitmq_federation和rabbitmq_shovel插件。

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
    .build()

try:
    request = UpdatePluginsRequest()
    request.instance_id = "{instance_id}"
    request.body = UpdatePluginsReq(
        plugins="rabbitmq_federation,rabbitmq_shovel",
        enable=True
    )
    response = client.update_plugins(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

开启rabbitmq_federation和rabbitmq_shovel插件。

```

package main

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

    request := &model.UpdatePluginsRequest{}
    request.InstanceId = "{instance_id}"
    pluginsUpdatePluginsReq:= "rabbitmq_federation,rabbitmq_shovel"
    enableUpdatePluginsReq:= true
    request.Body = &model.UpdatePluginsReq{
        Plugins: &pluginsUpdatePluginsReq,
    }
}

```

```
    Enable: &enableUpdatePluginsReq,  
  }  
  response, err := client.UpdatePlugins(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	开启或关闭插件成功。

错误码

请参见[错误码](#)。

5.3 规格变更管理

5.3.1 查询新规格可扩容规格列表

功能介绍

查询新规格实例可扩容列表

调用方法

请参见[如何调用API](#)。

URI

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

表 5-31 路径参数

参数	是否必选	参数类型	描述
engine	是	String	消息引擎的类型。支持的类型为rabbitmq。
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 5-32 Query 参数

参数	是否必选	参数类型	描述
type	否	String	产品的类型。 <ul style="list-style-type: none"> advanced: 专享版 platinum: 铂金版 dec: 专属云版 exp: 体验版

请求参数

无

响应参数

状态码: 200

表 5-33 响应 Body 参数

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

表 5-34 RabbitMQExtendProductInfoEntity

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。

参数	参数类型	描述
ios	Array of RabbitMQExtendProductIosEntity objects	磁盘IO信息。
properties	RabbitMQExtendProductPropertiesEntity object	功能特性的键值对。
available_zones	Array of strings	有可用资源的可用区列表。
unavailable_zones	Array of strings	资源售罄的可用区列表
support_features	Array of RabbitMQProductSupportFeaturesEntity objects	支持的特性功能。

表 5-35 RabbitMQExtendProductIosEntity

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 5-36 RabbitMQExtendProductPropertiesEntity

参数	参数类型	描述
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。
min_broker	String	Broker的最小个数。
min_storage_per_node	String	每个节点的最小存储。单位为GB。
max_connection_per_broker	String	最大连接数
step_length	String	步长
product_alias	String	product_id的别名。

参数	参数类型	描述
max_queue_per_broker	String	最大队列

表 5-37 RabbitMQProductSupportFeaturesEntity

参数	参数类型	描述
name	String	特性名称。
properties	Map<String,String>	功能特性的键值对。

请求示例

查询实例的扩容规格列表。

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

响应示例

状态码: 200

查询实例的扩容规格列表成功

```
{
  "engine": "rabbitmq",
  "versions": [ "3.8.35" ],
  "products": [ {
    "type": "single",
    "product_id": "c6.2u4g.single",
    "ecs_flavor_id": "c6.large.2",
    "arch_types": [ "X86" ],
    "charging_mode": [ "monthly", "hourly" ],
    "ios": [ {
      "io_spec": "dms.physical.storage.ultra.v2",
      "available_zones": [ "xxx" ],
      "type": "evs",
      "unavailable_zones": [ "xxx" ]
    }, {
      "io_spec": "dms.physical.storage.high.v2",
      "available_zones": [ "xxx" ],
      "type": "evs",
      "unavailable_zones": [ "xxx" ]
    } ],
    "support_features": [ ],
    "properties": {
      "max_connection_per_broker": "2000",
      "max_broker": "1",
      "max_queue_per_broker": "100",
      "max_storage_per_node": "30000",
      "min_broker": "1",
      "product_alias": "rabbitmq.2u4g.single",
      "step_length": "0",
      "min_storage_per_node": "100"
    }
  },
  "available_zones": [ "xxx" ],
  "unavailable_zones": [ ]
}
```

```

}, {
  "type": "single",
  "product_id": "c6.4u8g.single",
  "ecs_flavor_id": "c6.xlarge.2",
  "arch_types": [ "X86" ],
  "charging_mode": [ "monthly", "hourly" ],
  "ios": [ {
    "io_spec": "dms.physical.storage.high.v2",
    "available_zones": [ "xxx" ],
    "type": "evs",
    "unavailable_zones": [ "xxx" ]
  }, {
    "io_spec": "dms.physical.storage.ultra.v2",
    "available_zones": [ "xxx" ],
    "type": "evs",
    "unavailable_zones": [ "xxx" ]
  } ],
  "support_features": [ ],
  "properties": {
    "max_connection_per_broker": "3000",
    "max_broker": "1",
    "max_queue_per_broker": "200",
    "max_storage_per_node": "30000",
    "min_broker": "1",
    "product_alias": "rabbitmq.4u8g.single",
    "step_length": "0",
    "min_storage_per_node": "100"
  },
  "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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);
    }
}

```

```

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.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.getStatusCode());
    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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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](#)。

5.3.2 新规格实例的规格变更

功能介绍

实例规格变更。

当前通过调用API，只支持按需实例进行实例规格变更。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-38 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID，获取方式请参见 获取项目ID 。
instance_id	是	String	实例ID。
engine	是	String	消息引擎的类型。支持的类型为rabbitmq。

请求参数

表 5-39 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的底层虚拟机规格变更，代理数量和存储空间不变。

参数	是否必选	参数类型	描述
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_product_id	否	String	规格, 例如c6.8u16g.cluster, 当oper_type类型是vertical时, 该参数才有效且必填。
new_broker_num	否	Integer	当oper_type参数为horizontal时, 该参数有效。
new_spec_code	否	String	老规格, 例如dms.instance.rabbitmq.cluster.c3.8u16g, 当oper_type类型horizontal时, 为dms.instance.rabbitmq.cluster.c3.8u16g.5, 最后的数字5为代理数

响应参数

状态码: 200

表 5-40 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 (老规格按需实例)。new_spec_code的值为原规格参数。

```
POST https://{endpoint}/{engine}/v2/{project_id}/instances/{instance_id}/extend
{
  "new_storage_space": 600,
  "oper_type": "storage",
  "new_spec_code": "dms.instance.rabbitmq.cluster.c3.2u4g.3"
}
```

- 扩容代理数量（老规格按需实例）。new_storage_space填原存储大小，new_spec_code的值dms.instance.rabbitmq.cluster.c3.2u4g.5，最后数字5为代理个数。

```
POST https://{endpoint}/{engine}/v2/{project_id}/instances/{instance_id}/extend
{
  "new_storage_space": 600,
  "oper_type": "horizontal",
  "new_spec_code": "dms.instance.rabbitmq.cluster.c3.2u4g.5"
}
```

- 扩容代理规格（老规格按需实例）。new_spec_code需要改成对应规格，例如2u4g改成4u8g。

```
POST https://{endpoint}/{engine}/v2/{project_id}/instances/{instance_id}/extend
{
  "new_storage_space": 600,
  "oper_type": "vertical",
  "new_spec_code": "dms.instance.rabbitmq.cluster.c3.2u4g.5"
}
```

- 扩容存储空间（按需实例）。

```
POST https://{endpoint}/{engine}/v2/{project_id}/instances/{instance_id}/extend
{
  "new_storage_space": 600,
  "oper_type": "storage"
}
```

- 扩容代理数量（按需实例）。

```
POST https://{endpoint}/{engine}/v2/{project_id}/instances/{instance_id}/extend
{
  "oper_type": "horizontal",
  "new_storage_space": 500,
  "new_broker_num": 5
}
```

- 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

实例规格变更成功。

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

SDK 代码示例

SDK代码示例如下。

Java

- 扩容存储空间（老规格按需实例）。new_spec_code的值为原规格参数。

```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withInstanceId("{instance_id}");
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        body.withNewSpecCode("dms.instance.rabbitmq.cluster.c3.2u4g.5");
        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（老规格按需实例）。new_storage_space填原存储大小，new_spec_code的值dms.instance.rabbitmq.cluster.c3.2u4g.5，最后数字5为代理个数。

```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withInstanceId("{instance_id}");
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
        body.withNewSpecCode("dms.instance.rabbitmq.cluster.c3.2u4g.5");
        body.withNewStorageSpace(600);
        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（老规格按需实例）。new_spec_code需要改成对应规格，例如2u4g改成4u8g。

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
request.withInstanceId("{instance_id}");
request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
body.withNewSpecCode("dms.instance.rabbitmq.cluster.c3.2u4g.5");
body.withNewStorageSpace(600);
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());
}
}
```

- 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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
request.withInstanceId("{instance_id}");
request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withInstanceId("{instance_id}");
    }
}

```

```

request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
ResizeEngineInstanceReq body = new ResizeEngineInstanceReq();
body.withNewBrokerNum(5);
body.withNewStorageSpace(500);
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ResizeEngineInstanceRequest request = new ResizeEngineInstanceRequest();
        request.withInstanceId("{instance_id}");
        request.withEngine(ResizeEngineInstanceRequest.EngineEnum.fromValue("{engine}"));
        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（老规格按需实例）。new_spec_code的值为原规格参数。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeEngineInstanceRequest()
        request.instance_id = "{instance_id}"
        request.engine = "{engine}"
        request.body = ResizeEngineInstanceReq(
            new_spec_code="dms.instance.rabbitmq.cluster.c3.2u4g.3",
            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（老规格按需实例）。new_storage_space填原存储大小，new_spec_code的值dms.instance.rabbitmq.cluster.c3.2u4g.5，最后数字5为代理个数。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeEngineInstanceRequest()
        request.instance_id = "{instance_id}"
        request.engine = "{engine}"
        request.body = ResizeEngineInstanceReq(
            new_spec_code="dms.instance.rabbitmq.cluster.c3.2u4g.5",
            new_storage_space=600,
            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（老规格按需实例）。new_spec_code需要改成对应规格，例如2u4g改成4u8g。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeEngineInstanceRequest()
        request.instance_id = "{instance_id}"
        request.engine = "{engine}"
        request.body = ResizeEngineInstanceReq(

```

```

        new_spec_code="dms.instance.rabbitmq.cluster.c3.2u4g.5",
        new_storage_space=600,
        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)

```

- 扩容存储空间（按需实例）。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeEngineInstanceRequest()
        request.instance_id = "{instance_id}"
        request.engine = "{engine}"
        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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
.with_credentials(credentials) \
.with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
.build()

try:
request = ResizeEngineInstanceRequest()
request.instance_id = "{instance_id}"
request.engine = "{engine}"
request.body = ResizeEngineInstanceReq(
    new_broker_num=5,
    new_storage_space=500,
    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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ResizeEngineInstanceRequest()
        request.instance_id = "{instance_id}"
        request.engine = "{engine}"
        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（老规格按需实例）。new_spec_code的值为原规格参数。

```
package main

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

    request := &model.ResizeEngineInstanceRequest{}
    request.InstanceId = "{instance_id}"
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    newSpecCodeResizeEngineInstanceReq:= "dms.instance.rabbitmq.cluster.c3.2u4g.3"
    newStorageSpaceResizeEngineInstanceReq:= int32(600)
    request.Body = &model.ResizeEngineInstanceReq{
        NewSpecCode: &newSpecCodeResizeEngineInstanceReq,
        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（老规格按需实例）。new_storage_space填原存储大小，new_spec_code的值为dms.instance.rabbitmq.cluster.c3.2u4g.5，最后数字5为代理个数。

```
package main

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

    request := &model.ResizeEngineInstanceRequest{}
    request.InstanceId = "{instance_id}"
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    newSpecCodeResizeEngineInstanceReq:= "dms.instance.rabbitmq.cluster.c3.2u4g.5"
    newStorageSpaceResizeEngineInstanceReq:= int32(600)
    request.Body = &model.ResizeEngineInstanceReq{
        NewSpecCode: &newSpecCodeResizeEngineInstanceReq,
        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（老规格按需实例）。new_spec_code需要改成对应规格，例如2u4g改成4u8g。

```

package main

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

request := &model.ResizeEngineInstanceRequest{}
request.InstanceId = "{instance_id}"
request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
newSpecCodeResizeEngineInstanceReq:= "dms.instance.rabbitmq.cluster.c3.2u4g.5"
newStorageSpaceResizeEngineInstanceReq:= int32(600)
request.Body = &model.ResizeEngineInstanceReq{
    NewSpecCode: &newSpecCodeResizeEngineInstanceReq,
    NewStorageSpace: &newStorageSpaceResizeEngineInstanceReq,
    OperType: "vertical",
}
response, err := client.ResizeEngineInstance(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
}

```

- 扩容存储空间（按需实例）。

```

package main

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

    request := &model.ResizeEngineInstanceRequest{}
    request.InstanceId = "{instance_id}"
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResizeEngineInstanceRequest{}
    request.InstanceId = "{instance_id}"
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    newBrokerNumResizeEngineInstanceReq:= int32(5)
    newStorageSpaceResizeEngineInstanceReq:= int32(500)
    request.Body = &model.ResizeEngineInstanceReq{
        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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ResizeEngineInstanceRequest{}
    request.InstanceId = "{instance_id}"
    request.Engine = model.GetResizeEngineInstanceRequestEngineEnum().ENGINE
    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	实例规格变更成功。

错误码

请参见[错误码](#)。

5.4 Vhost 管理

5.4.1 创建 Vhost

功能介绍

创建Vhost。

调用方法

请参见[如何调用API](#)。

URI

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

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
name	是	String	Vhost名称

响应参数

无

请求示例

创建Vhost

```
PUT https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts
{
  "name": "vhost-demo"
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

创建Vhost

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class CreateVhostSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreateVhostRequest request = new CreateVhostRequest();
        request.withInstanceId("{instance_id}");
        CreateVhostBody body = new CreateVhostBody();
        body.withName("vhost-demo");
        request.withBody(body);
        try {
            CreateVhostResponse response = client.createVhost(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Vhost

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateVhostRequest()
        request.instance_id = "{instance_id}"
        request.body = CreateVhostBody(
            name="vhost-demo"
        )
        response = client.create_vhost(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Vhost

```
package main

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

request := &model.CreateVhostRequest{}
request.InstanceId = "{instance_id}"
request.Body = &model.CreateVhostBody{
    Name: "vhost-demo",
}
response, err := client.CreateVhost(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
```

更多

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

状态码

状态码	描述
200	创建成功

错误码

请参见[错误码](#)。

5.4.2 查询 Vhost 列表

功能介绍

查询Vhost列表。

调用方法

请参见[如何调用API](#)。

URI

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

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 。

参数	是否必选	参数类型	描述
instance_id	是	String	实例ID

表 5-44 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	分页查询偏移量，表示从此偏移量开始查询，offset大于等于0，默认从0开始查询。
limit	否	Integer	分页查询单页数量，取值范围0~50，默认查询10条。

请求参数

无

响应参数

状态码：200

表 5-45 响应 Body 参数

参数	参数类型	描述
size	Integer	当前显示数量
total	Integer	查询结果总数
items	Array of ShowVhostDetailResp objects	查询的Vhost信息列表

表 5-46 ShowVhostDetailResp

参数	参数类型	描述
name	String	Vhost名称。
tracing	Boolean	是否开启消息轨迹（AMQP版本不涉及此字段）。

请求示例

查询Vhost列表

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

响应示例

状态码: 200

获取Vhost列表成功

```
{
  "size" : 10,
  "total" : 13,
  "items" : [ {
    "name" : "/",
    "tracing" : false
  }, {
    "name" : "test-vhost1",
    "tracing" : false
  }, {
    "name" : "test-vhost10",
    "tracing" : false
  }, {
    "name" : "test-vhost2",
    "tracing" : false
  }, {
    "name" : "test-vhost3",
    "tracing" : false
  }, {
    "name" : "test-vhost4",
    "tracing" : false
  }, {
    "name" : "test-vhost5",
    "tracing" : false
  }, {
    "name" : "test-vhost6",
    "tracing" : false
  }, {
    "name" : "test-vhost7",
    "tracing" : false
  }, {
    "name" : "test-vhost8",
    "tracing" : 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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListVhostsSolution {

    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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
ListVhostsRequest request = new ListVhostsRequest();
request.withInstanceId("{instance_id}");
try {
    ListVhostsResponse response = client.listVhosts(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

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

    try:
        request = ListVhostsRequest()
        request.instance_id = "{instance_id}"
        response = client.list_vhosts(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListVhostsRequest{}
    request.InstanceId = "{instance_id}"
    response, err := client.ListVhosts(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Vhost列表成功

错误码

请参见[错误码](#)。

5.4.3 批量删除指定 Vhost

功能介绍

批量删除指定Vhost。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts

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

表 5-48 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	Array of strings	需要删除的资源名称列表

响应参数

无

请求示例

批量删除指定Vhost

```
POST https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts
{
  "name" : [ "vhost1", "vhost2" ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

批量删除指定Vhost

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class BatchDeleteVhostsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        BatchDeleteVhostsRequest request = new BatchDeleteVhostsRequest();
        request.withInstanceId("{instance_id}");
        BatchDeleteBody body = new BatchDeleteBody();
        List<String> listbodyName = new ArrayList<>();
        listbodyName.add("vhost1");
        listbodyName.add("vhost2");
        body.withName(listbodyName);
        request.withBody(body);
        try {
            BatchDeleteVhostsResponse response = client.batchDeleteVhosts(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

批量删除指定Vhost

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchDeleteVhostsRequest()
        request.instance_id = "{instance_id}"
        listNamebody = [
            "vhost1",
            "vhost2"
        ]
        request.body = BatchDeleteBody(
            name=listNamebody
        )
        response = client.batch_delete_vhosts(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

批量删除指定Vhost

```
package main

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

request := &model.BatchDeleteVhostsRequest{}
request.InstanceId = "{instance_id}"
var listNamebody = []string{
    "vhost1",
    "vhost2",
}
request.Body = &model.BatchDeleteBody{
    Name: listNamebody,
}
response, err := client.BatchDeleteVhosts(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	删除成功

错误码

请参见[错误码](#)。

5.5 Exchange 管理

5.5.1 创建 Exchange

功能介绍

创建Exchange。

调用方法

请参见[如何调用API](#)。

URI

PUT /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges

表 5-49 路径参数

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

请求参数

表 5-50 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	String	Exchange名称
type	是	String	类型 (direct、fanout、topic、headers)
durable	否	Boolean	是否持久化 (AMQP版本默认持久化, 不涉及此参数)。
auto_delete	是	Boolean	是否自动删除
internal	否	Boolean	内部Exchange (AMQP版本不支持内部Exchange, 不涉及此参数)。

响应参数

状态码: 200

表 5-51 响应 Body 参数

参数	参数类型	描述
durable	Boolean	是否持久化
default	Boolean	是否是默认Exchange

参数	参数类型	描述
internal	Boolean	是否是内部Exchange
name	String	Exchange名称
auto_delete	Boolean	是否自动删除
type	String	Exchange类型
vhost	String	所属Vhost

请求示例

创建Exchange

POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges

```
{
  "name" : "exchange_name_demo",
  "type" : "direct",
  "durable" : true,
  "auto_delete" : false,
  "internal" : false
}
```

响应示例

状态码: 200

创建Exchange成功

```
{
  "name" : "exchange_name_demo",
  "type" : "direct",
  "durable" : true,
  "auto_delete" : false,
  "internal" : false,
  "vhost" : "default"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建Exchange

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;
```

```
public class CreateExchangeSolution {
    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreateExchangeRequest request = new CreateExchangeRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        CreateExchangeBody body = new CreateExchangeBody();
        body.withInternal(false);
        body.withAutoDelete(false);
        body.withDurable(true);
        body.withType("direct");
        body.withName("exchange_name_demo");
        request.withBody(body);
        try {
            CreateExchangeResponse response = client.createExchange(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.getStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

创建Exchange

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdbkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdbkrabbitmq.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")
    projectId = "{project_id}"
```



```

credentials = BasicCredentials(ak, sk, projectId)

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

try:
    request = CreateExchangeRequest()
    request.instance_id = "{instance_id}"
    request.vhost = "{vhost}"
    request.body = CreateExchangeBody(
        internal=False,
        auto_delete=False,
        durable=True,
        type="direct",
        name="exchange_name_demo"
    )
    response = client.create_exchange(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Exchange

```

package main

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

    request := &model.CreateExchangeRequest{}
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    internalCreateExchangeBody:= false
    durableCreateExchangeBody:= true
    request.Body = &model.CreateExchangeBody{

```

```
Internal: &internalCreateExchangeBody,  
AutoDelete: false,  
Durable: & durableCreateExchangeBody,  
Type: "direct",  
Name: "exchange_name_demo",  
}  
response, err := client.CreateExchange(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Exchange成功

错误码

请参见[错误码](#)。

5.5.2 查询 Exchange 列表

功能介绍

查询Exchange列表。

调用方法

请参见[如何调用API](#)。

URI

GET /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges

表 5-52 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
vhost	是	String	所属Vhost名称

表 5-53 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	分页查询偏移量，表示从此偏移量开始查询，offset大于等于0，默认从0开始查询。
limit	否	Integer	分页查询单页数量，取值范围0~50，默认查询10条。

请求参数

无

响应参数

状态码：200

表 5-54 响应 Body 参数

参数	参数类型	描述
size	Integer	当前显示数量
total	Integer	查询结果总数
items	Array of ExchangeDetails objects	Exchange信息列表

表 5-55 ExchangeDetails

参数	参数类型	描述
durable	Boolean	是否持久化
default	Boolean	是否是默认Exchange
internal	Boolean	是否是内部Exchange
name	String	Exchange名称
auto_delete	Boolean	是否自动删除
type	String	Exchange类型
vhost	String	所属Vhost

请求示例

查询Exchange列表

```
GET /v2/rabbitmq/{project_id}/instances/{instance_id}/exchanges?offset=0&limit=10
```

响应示例

状态码: 200

获取Exchange列表成功

```
{
  "total": 1,
  "size": 1,
  "items": [ {
    "durable": true,
    "default": false,
    "internal": false,
    "name": "default",
    "auto_delete": false,
    "type": "direct"
  } ]
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListExchangesSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ListExchangesRequest request = new ListExchangesRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        try {
            ListExchangesResponse response = client.listExchanges(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListExchangesRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        response = client.list_exchanges(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        WithRegion(region.ValueOf("<YOUR REGION>")).
        WithCredential(auth).
        Build())

request := &model.ListExchangesRequest{}
request.InstanceId = "{instance_id}"
request.Vhost = "{vhost}"
response, err := client.ListExchanges(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Exchange列表成功

错误码

请参见[错误码](#)。

5.5.3 批量删除指定 Exchange

功能介绍

批量删除指定Exchange。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges

表 5-56 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
vhost	是	String	Vhost名称

请求参数

表 5-57 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	Array of strings	需要删除的资源名称列表

响应参数

无

请求示例

批量删除指定Exchange

```
POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges
{
  "name": [ "exchange1", "exchange2" ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

批量删除指定Exchange

```
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.rabbitmq.v2.region.RabbitMQRegion;
```

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

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

public class BatchDeleteExchangesSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        BatchDeleteExchangesRequest request = new BatchDeleteExchangesRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        BatchDeleteBody body = new BatchDeleteBody();
        List<String> listbodyName = new ArrayList<>();
        listbodyName.add("exchange1");
        listbodyName.add("exchange2");
        body.withName(listbodyName);
        request.withBody(body);
        try {
            BatchDeleteExchangesResponse response = client.batchDeleteExchanges(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.getStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

Python

批量删除指定Exchange

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
    .build()

try:
    request = BatchDeleteExchangesRequest()
    request.instance_id = "{instance_id}"
    request.vhost = "{vhost}"
    listNamebody = [
        "exchange1",
        "exchange2"
    ]
    request.body = BatchDeleteBody(
        name=listNamebody
    )
    response = client.batch_delete_exchanges(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

批量删除指定Exchange

```
package main

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

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

```
request.InstanceId = "{instance_id}"
request.Vhost = "{vhost}"
var listNamebody = []string{
    "exchange1",
    "exchange2",
}
request.Body = &model.BatchDeleteBody{
    Name: listNamebody,
}
response, err := client.BatchDeleteExchanges(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	删除成功

错误码

请参见[错误码](#)。

5.6 Queue 管理

5.6.1 创建 Queue

功能介绍

创建Queue。

调用方法

请参见[如何调用API](#)。

URI

PUT /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues

表 5-58 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
vhost	是	String	Vhost名称

请求参数

表 5-59 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	String	Queue名称
auto_delete	是	Boolean	是否自动删除
durable	否	Boolean	是否持久化 (AMQP版本默认持久化, 不涉及此字段)
dead_letter_exchange	否	String	死信Exchange名称, 消息被拒绝或过期时将重新发布到该Exchange。
dead_letter_routing_key	否	String	死信Exchange的RoutingKey, 死信Exchange会发送死信消息到绑定对应RoutingKey的Queue上。
message_ttl	否	Long	发布到Queue的消息在被丢弃之前可以存活多长时间
lazy_mode	否	String	若设置惰性队列, 请输入lazy。惰性队列模式会在磁盘上存储尽可能多的消息以减少内存使用; 若不设置, 队列将消息存储在内存缓存以尽可能快地传递消息。(AMQP版本默认将消息存储到磁盘, 不涉及此字段)

响应参数

状态码: 200

表 5-60 响应 Body 参数

参数	参数类型	描述
name	String	Queue名称
auto_delete	Boolean	是否自动删除
durable	Boolean	是否持久化（AMQP版本默认持久化，不涉及此字段）
dead_letter_exchange	String	死信Exchange名称，消息被拒绝或过期时将重新发布到该Exchange。
dead_letter_routing_key	String	死信Exchange的RoutingKey，死信Exchange会发送死信消息到绑定对应RoutingKey的Queue上。
message_ttl	Long	发布到Queue的消息在被丢弃之前可以存活多长时间
lazy_mode	String	若设置惰性队列，请输入lazy。惰性队列模式会在磁盘上存储尽可能多的消息以减少内存使用；若不设置，队列将消息存储在内存缓存以尽可能快地传递消息。（AMQP版本默认将消息存储到磁盘，不涉及此字段）

请求示例

创建Queue

PUT https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues

```
{
  "name": "string",
  "auto_delete": true,
  "durable": true,
  "dead_letter_exchange": "string",
  "dead_letter_routing_key": "string",
  "message_ttl": 6000,
  "lazy_mode": "string"
}
```

响应示例

状态码: 200

创建Queue成功

```
{
  "name": "string",
  "auto_delete": true,
  "durable": true,
  "dead_letter_exchange": "string",
  "dead_letter_routing_key": "string",
  "message_ttl": 60000,
  "lazy_mode": "string"
}
```

SDK 代码示例

SDK代码示例如下。

Java

创建Queue

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class CreateQueueSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreateQueueRequest request = new CreateQueueRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        CreateQueueBody body = new CreateQueueBody();
        body.withLazyMode("string");
        body.withMessageTtl(6000L);
        body.withDeadLetterRoutingKey("string");
        body.withDeadLetterExchange("string");
        body.withDurable(true);
        body.withAutoDelete(true);
        body.withName("string");
        request.withBody(body);
        try {
            CreateQueueResponse response = client.createQueue(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.getStatusCode());
            System.out.println(e.getRequestId());
            System.out.println(e.getErrorCode());
            System.out.println(e.getErrorMsg());
        }
    }
}
```

```
}  
}
```

Python

创建Queue

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateQueueRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        request.body = CreateQueueBody(
            lazy_mode="string",
            message_ttl=6000,
            dead_letter_routing_key="string",
            dead_letter_exchange="string",
            durable=True,
            auto_delete=True,
            name="string"
        )
        response = client.create_queue(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Queue

```
package main

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

request := &model.CreateQueueRequest{
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    lazyModeCreateQueueBody:= "string"
    messageTtlCreateQueueBody:= int64(6000)
    deadLetterRoutingKeyCreateQueueBody:= "string"
    deadLetterExchangeCreateQueueBody:= "string"
    durableCreateQueueBody:= true
    request.Body = &model.CreateQueueBody{
        LazyMode: &lazyModeCreateQueueBody,
        MessageTtl: &messageTtlCreateQueueBody,
        DeadLetterRoutingKey: &deadLetterRoutingKeyCreateQueueBody,
        DeadLetterExchange: &deadLetterExchangeCreateQueueBody,
        Durable: &durableCreateQueueBody,
        AutoDelete: true,
        Name: "string",
    }
}
response, err := client.CreateQueue(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Queue成功

错误码

请参见[错误码](#)。

5.6.2 查询所属 Vhost 下 Queue 的列表

功能介绍

查询所属Vhost下Queue的列表。

调用方法

请参见[如何调用API](#)。

URI

GET /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues

表 5-61 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
vhost	是	String	Vhost名称

表 5-62 Query 参数

参数	是否必选	参数类型	描述
offset	否	Integer	分页查询偏移量, 表示从此偏移量开始查询, offset大于等于0, 默认从0开始查询。
limit	否	Integer	分页查询单页数量, 取值范围0~50, 默认查询10条。

请求参数

无

响应参数

状态码: 200

表 5-63 响应 Body 参数

参数	参数类型	描述
size	Integer	当前显示数量

参数	参数类型	描述
total	Integer	查询结果总数
items	Array of QueueDetails objects	查询详情

表 5-64 QueueDetails

参数	参数类型	描述
vhost	String	Queue所属Vhost名称
name	String	Queue名称
durable	Boolean	是否持久化
auto_delete	Boolean	是否自动删除
messages	Integer	待消费消息数
consumers	Integer	连接的消费者数
policy	String	策略 (AMQP版本不支持policy, 不涉及此参数)
arguments	QueueArguments object	Queue参数, 如果未配置则不返回。

表 5-65 QueueArguments

参数	参数类型	描述
x-message-ttl	Long	消息过期时间, 发布到Queue的消息在被丢弃之前可以存活多长时间。
x-dead-letter-exchange	String	死信Exchange名称, 消息被拒绝或过期时将重新发布到该Exchange。
x-dead-letter-routing-key	String	死信的RoutingKey, 死信Exchange会发送死信消息到绑定对应RoutingKey的Queue上。
x-queue-mode	String	惰性队列 (AMQP版本默认持久化所有消息, 不涉及此参数)

请求示例

查询Queue列表

```
GET https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues?
offset=0&limit=10
```

响应示例

状态码: 200

获取Queue列表成功

```
{
  "size" : 1,
  "total" : 1,
  "items" : [ {
    "durable" : true,
    "name" : "queue10",
    "auto_delete" : false,
    "messages" : 0,
    "consumers" : 0,
    "arguments" : {
      "x-dead-letter-exchange" : "dead-exchange-deal",
      "x-dead-letter-routing-key" : "dead-ex-routing-key",
      "x-message-ttl" : 60000
    }
  }
  ]
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListQueuesSolution {

    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);

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

```

request.withVhost("{vhost}");
try {
    ListQueuesResponse response = client.listQueues(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListQueuesRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        response = client.list_queues(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListQueuesRequest{}
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    response, err := client.ListQueues(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Queue列表成功

错误码

请参见[错误码](#)。

5.6.3 批量删除指定 Queue

功能介绍

批量删除指定Queue。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues

表 5-66 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
vhost	是	String	Vhost名称

请求参数

表 5-67 请求 Body 参数

参数	是否必选	参数类型	描述
name	是	Array of strings	需要删除的资源名称列表

响应参数

无

请求示例

批量删除Queue

```
POST https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues
{
  "name" : [ "queue1", "queue2" ]
}
```

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

批量删除Queue

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class BatchDeleteQueuesSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        BatchDeleteQueuesRequest request = new BatchDeleteQueuesRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        BatchDeleteBody body = new BatchDeleteBody();
        List<String> listbodyName = new ArrayList<>();
        listbodyName.add("queue1");
        listbodyName.add("queue2");
        body.withName(listbodyName);
        request.withBody(body);
        try {
            BatchDeleteQueuesResponse response = client.batchDeleteQueues(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Queue

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = BatchDeleteQueuesRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        listNamebody = [
            "queue1",
            "queue2"
        ]
        request.body = BatchDeleteBody(
            name=listNamebody
        )
        response = client.batch_delete_queues(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Queue

```

package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().

```

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

request := &model.BatchDeleteQueuesRequest{}
request.InstanceId = "{instance_id}"
request.Vhost = "{vhost}"
var listNamebody = []string{
    "queue1",
    "queue2",
}
request.Body = &model.BatchDeleteBody{
    Name: listNamebody,
}
response, err := client.BatchDeleteQueues(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	删除成功

错误码

请参见[错误码](#)。

5.6.4 清空 Queue 消息

功能介绍

清空Queue消息。

调用方法

请参见[如何调用API](#)。

URI

DELETE /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues/{queue}/contents

表 5-68 路径参数

参数	是否必选	参数类型	描述
project_id	是	String	项目ID, 获取方式请参见 获取项目ID 。
instance_id	是	String	实例ID
vhost	是	String	Vhost名称
queue	是	String	Queue名称

请求参数

无

响应参数

无

请求示例

清空Queue消息

```
DELETE /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues/{queue}/contents
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class DeleteQueueInfoSolution {

    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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
DeleteQueueInfoRequest request = new DeleteQueueInfoRequest();
request.withInstanceId("{instance_id}");
request.withVhost("{vhost}");
request.withQueue("{queue}");
try {
    DeleteQueueInfoResponse response = client.deleteQueueInfo(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteQueueInfoRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        request.queue = "{queue}"
        response = client.delete_queue_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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.DeleteQueueInfoRequest{}
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    request.Queue = "{queue}"
    response, err := client.DeleteQueueInfo(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	清空Queue消息成功

错误码

请参见[错误码](#)。

5.6.5 查询指定 Queue 详情

功能介绍

查询指定Queue详情。

调用方法

请参见[如何调用API](#)。

URI

GET /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues/{queue}

表 5-69 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
vhost	是	String	所属Vhost名称
queue	是	String	Queue名称

请求参数

无

响应参数

状态码: 200

表 5-70 响应 Body 参数

参数	参数类型	描述
vhost	String	Queue所属Vhost名称
name	String	Queue名称
durable	Boolean	是否持久化
auto_delete	Boolean	是否自动删除
messages	Integer	待消费消息数
consumers	Integer	连接的消费者数

参数	参数类型	描述
policy	String	策略 (AMQP版本不支持policy, 不涉及此参数)
arguments	QueueArguments object	Queue参数, 如果未配置则不返回。
consumer_details	Array of ConsumerDetails objects	订阅该Queue的消费者信息。
queue_bindings	Array of BindingsDetails objects	以此Queue为目标的绑定信息列表。

表 5-71 QueueArguments

参数	参数类型	描述
x-message-ttl	Long	消息过期时间, 发布到Queue的消息在被丢弃之前可以存活多长时间。
x-dead-letter-exchange	String	死信Exchange名称, 消息被拒绝或过期时将重新发布到该Exchange。
x-dead-letter-routing-key	String	死信的RoutingKey, 死信Exchange会发送死信消息到绑定对应RoutingKey的Queue上。
x-queue-mode	String	惰性队列 (AMQP版本默认持久化所有消息, 不涉及此参数)

表 5-72 ConsumerDetails

参数	参数类型	描述
consumer_tag	String	消费者标识
channel_details	ChannelDetails object	消费者连接信息
ack_required	Boolean	消费者客户端是否设置手动ack
prefetch_count	Integer	消费者客户端预取值

表 5-73 ChannelDetails

参数	参数类型	描述
name	String	channel信息, 包括客户端IP:Port到服务端IP:Port(channel_id)。
number	Integer	channel数量
user	String	消费者用户名, 在开启ACL访问控制后返回真实用户名, 未开启ACL时返回null。
connection_name	String	connection信息, 包括客户端IP:Port到服务端IP:Port。
peer_host	String	连接的消费者IP
peer_port	Integer	连接的消费者进程端口号

表 5-74 BindingsDetails

参数	参数类型	描述
source	String	Exchange名称
destination_type	String	绑定目标类型
destination	String	绑定目标的名称
routing_key	String	绑定键值
properties_key	String	经过URL转译后routing_key

请求示例

查询指定Queue详情

```
GET https://{endpoint}/v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/queues?offset=0&limit=10
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ShowQueueDetailsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowQueueDetailsRequest request = new ShowQueueDetailsRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        request.withQueue("{queue}");
        try {
            ShowQueueDetailsResponse response = client.showQueueDetails(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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")
    projectId = "{project_id}"
```

```

credentials = BasicCredentials(ak, sk, projectId)

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

try:
    request = ShowQueueDetailsRequest()
    request.instance_id = "{instance_id}"
    request.vhost = "{vhost}"
    request.queue = "{queue}"
    response = client.show_queue_details(request)
    print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)

```

Go

```

package main

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

    request := &model.ShowQueueDetailsRequest{}
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    request.Queue = "{queue}"
    response, err := client.ShowQueueDetails(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Queue详情成功

错误码

请参见[错误码](#)。

5.7 Binding 管理

5.7.1 添加绑定

功能介绍

添加绑定。

调用方法

请参见[如何调用API](#)。

URI

POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges/{exchange}/binding

表 5-75 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 。
instance_id	是	String	实例ID
vhost	是	String	Vhost名称
exchange	是	String	Exchange名称

请求参数

表 5-76 请求 Body 参数

参数	是否必选	参数类型	描述
destination	是	String	要投递的Exchange或Queue名称
routing_key	是	String	绑定键值，用于告知Exchange应该将消息投递到哪些Queue或Exchange中
destination_type	是	String	绑定目标端类型，Exchange或Queue。（AMQP版本只支持绑定Queue）

响应参数

状态码：200

表 5-77 响应 Body 参数

参数	参数类型	描述
source	String	绑定对象
destination_type	String	绑定Exchange或者Queue
destination	String	要投递的Exchange或Queue名称
routing_key	String	绑定键值，用于告知Exchange应该将消息投递到哪些Queue中

请求示例

将test-exchange作为源端，与目标端为Queue类型的mirror-queue绑定，绑定路由键为routing_key_1。

```
POST /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/default/exchanges/test-exchange/binding
{
  "destination_type": "Queue",
  "destination": "mirror-queue",
  "routing_key": "routing_key_1"
}
```

响应示例

状态码：200

添加绑定成功

```
{
  "source": "exchange_name",
```

```
"destination_type": "Queue",
"destination": "queue_name",
"routing_key": "binding_key_demo"
}
```

SDK 代码示例

SDK代码示例如下。

Java

将test-exchange作为源端，与目标端为Queue类型的mirror-queue绑定，绑定路由键为routing_key_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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class CreateBindingSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        CreateBindingRequest request = new CreateBindingRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        request.withExchange("{exchange}");
        CreateBindingBody body = new CreateBindingBody();
        body.withDestinationType("Queue");
        body.withRoutingKey("routing_key_1");
        body.withDestination("mirror-queue");
        request.withBody(body);
        try {
            CreateBindingResponse response = client.createBinding(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

将test-exchange作为源端，与目标端为Queue类型的mirror-queue绑定，绑定路由键为routing_key_1。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateBindingRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        request.exchange = "{exchange}"
        request.body = CreateBindingBody(
            destination_type="Queue",
            routing_key="routing_key_1",
            destination="mirror-queue"
        )
        response = client.create_binding(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

将test-exchange作为源端，与目标端为Queue类型的mirror-queue绑定，绑定路由键为routing_key_1。

```

package main

import (
    "fmt"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/core/auth/basic"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(  
        rabbitmq.RabbitMQClientBuilder().  
            WithRegion(region.ValueOf("<YOUR REGION>")).  
            WithCredential(auth).  
            Build())  
  
    request := &model.CreateBindingRequest{}  
    request.InstanceId = "{instance_id}"  
    request.Vhost = "{vhost}"  
    request.Exchange = "{exchange}"  
    request.Body = &model.CreateBindingBody{  
        DestinationType: "Queue",  
        RoutingKey: "routing_key_1",  
        Destination: "mirror-queue",  
    }  
    response, err := client.CreateBinding(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	添加绑定成功

错误码

请参见[错误码](#)。

5.7.2 查询 Exchange 绑定信息列表

功能介绍

查询Exchange绑定信息列表。

调用方法

请参见[如何调用API](#)。

URI

GET /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges/{exchange}/binding

表 5-78 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
vhost	是	String	Vhost名称
exchange	是	String	Exchange名称

请求参数

无

响应参数

状态码: 200

表 5-79 响应 Body 参数

参数	参数类型	描述
size	Integer	当前显示数量
total	Integer	查询结果总数
items	Array of BindingsDetails objects	绑定信息列表

表 5-80 BindingsDetails

参数	参数类型	描述
source	String	Exchange名称
destination_type	String	绑定目标类型
destination	String	绑定目标的名称

参数	参数类型	描述
routing_key	String	绑定键值
properties_key	String	经过URL转译后routing_key

请求示例

查询Exchange绑定信息列表

```
GET /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges/{exchange}/binding
```

响应示例

状态码：200

查询Exchange绑定信息列表成功

```
{
  "size" : 1,
  "total" : 1,
  "items" : [ {
    "source" : "excahnge-test",
    "destination_type" : "queue",
    "destination" : "queue-test",
    "routing_key" : "test-routing-key",
    "properties_key" : "test-routing-key"
  } ]
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ListBindingsSolution {

    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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
ListBindingsRequest request = new ListBindingsRequest();
request.withInstanceId("{instance_id}");
request.withVhost("{vhost}");
request.withExchange("{exchange}");
try {
    ListBindingsResponse response = client.listBindings(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.getStatusCode());
    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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ListBindingsRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        request.exchange = "{exchange}"
        response = client.list_bindings(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ListBindingsRequest{}
    request.InstanceId = "{instance_id}"
    request.Vhost = "{vhost}"
    request.Exchange = "{exchange}"
    response, err := client.ListBindings(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Exchange绑定信息列表成功

错误码

请参见[错误码](#)。

5.7.3 删除绑定

功能介绍

删除绑定。

调用方法

请参见[如何调用API](#)。

URI

DELETE /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/
exchanges/{exchange}/destination-type/{destination_type}/destination/
{destination}/properties-key/{properties_key}/unbinding

表 5-81 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 。
instance_id	是	String	实例ID
vhost	是	String	Vhost名称
exchange	是	String	Exchange名称
destination_type	是	String	绑定目标端类型, Exchange或Queue。(AMQP版本只支持Queue绑定类型)
destination	是	String	绑定的目标端名称
properties_key	是	String	绑定路由键, 经过URL转译后routing_key, 可通过调用 查询Exchange绑定列表 或者 查询指定Queue详情 接口的响应信息获取。

请求参数

无

响应参数

无

请求示例

解除绑定

```
DELETE /v2/rabbitmq/{project_id}/instances/{instance_id}/vhosts/{vhost}/exchanges/{exchange}/destination-type/{destination_type}/destination/{destination}/properties-key/{properties_key}/unbinding
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class DeleteBindingSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        DeleteBindingRequest request = new DeleteBindingRequest();
        request.withInstanceId("{instance_id}");
        request.withVhost("{vhost}");
        request.withExchange("{exchange}");
        request.withDestinationType("{destination_type}");
        request.withDestination("{destination}");
        request.withPropertiesKey("{properties_key}");
        try {
            DeleteBindingResponse response = client.deleteBinding(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.getStatusCode());
            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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteBindingRequest()
        request.instance_id = "{instance_id}"
        request.vhost = "{vhost}"
        request.exchange = "{exchange}"
        request.destination_type = "{destination_type}"
        request.destination = "{destination}"
        request.properties_key = "{properties_key}"
        response = client.delete_binding(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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        WithRegion(region.ValueOf("<YOUR REGION>")).
        WithCredential(auth).
        Build())

request := &model.DeleteBindingRequest{}
request.InstanceId = "{instance_id}"
request.Vhost = "{vhost}"
request.Exchange = "{exchange}"
request.DestinationType = "{destination_type}"
request.Destination = "{destination}"
request.PropertiesKey = "{properties_key}"
response, err := client.DeleteBinding(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	解绑成功

错误码

请参见[错误码](#)。

5.8 用户管理

5.8.1 创建用户

功能介绍

创建用户（仅AMQP版本支持）。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-82 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 5-83 请求 Body 参数

参数	是否必选	参数类型	描述
access_key	否	String	用户名, 只能英文字母开头, 且由英文字母、数字、中划线、下划线组成, 长度为7~64个字符。
secret_key	否	String	密钥。 8-32个字符。 至少包含以下字符中的3种: <ul style="list-style-type: none"> • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#%&*()-_=+ \[{}];:","<.>/?。 不能与名称或倒序的名称相同。
vhosts	否	Array of AMQPUserPerm objects	需要配置权限的Vhost, 一个用户可以配置多个Vhost下的资源权限。

表 5-84 AMQPUserPerm

参数	是否必选	参数类型	描述
vhost	否	String	需要配置权限的Vhost名称, 一个用户可以配置多个Vhost下的资源权限。

参数	是否必选	参数类型	描述
conf	否	String	使用正则表达式匹配资源配置权限。例如，在输入框内输入“^janeway-*”，则表示授权给该用户当前Vhost下，所有名称以“janeway-”开头的资源的配置权限。
write	否	String	使用正则表达式匹配资源写权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的写权限。
read	否	String	使用正则表达式匹配资源读权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的读权限。

响应参数

状态码：200

表 5-85 响应 Body 参数

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： <ul style="list-style-type: none"> • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#%&*()-_+=+ [{}];:","<>/?。 不能与名称或倒序的名称相同。
vhosts	Array of AMQPUserPerm objects	需要配置权限的Vhost，一个用户可以配置多个Vhost下的资源权限。

表 5-86 AMQPUserPerm

参数	参数类型	描述
vhost	String	需要配置权限的Vhost名称，一个用户可以配置多个Vhost下的资源权限。
conf	String	使用正则表达式匹配资源配置权限。例如，在输入框内输入“^janeway-.*”，则表示授权给该用户当前Vhost下，所有名称以“janeway-”开头的资源的配置权限。
write	String	使用正则表达式匹配资源写权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的写权限。
read	String	使用正则表达式匹配资源读权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的读权限。

请求示例

创建一个AMQP用户，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行配置、读、写。

POST https://{endpoint}/v2/{project_id}/instances/{instance_id}/users

```
{
  "access_key": "admin123",
  "secret_key": "*****",
  "vhosts": [ {
    "vhost": "default",
    "conf": ".*",
    "write": ".*",
    "read": ".*"
  } ]
}
```

响应示例

状态码：200

创建用户成功。

```
{
  "access_key": "admin123",
  "secret_key": "*****",
  "vhosts": [ {
    "vhost": "default",
    "conf": ".*",
    "write": ".*",
    "read": ".*"
  } ]
}
```


SDK 代码示例

SDK代码示例如下。

Java

创建一个AMQP用户，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行配置、读、写。

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class CreateUserSolution {

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

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

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();

        CreateUserRequest request = new CreateUserRequest();
        request.withInstanceId("{instance_id}");
        AMQPUser body = new AMQPUser();
        List<AMQPUserPerm> listbodyVhosts = new ArrayList<>();
        listbodyVhosts.add(
            new AMQPUserPerm()
                .withVhost("default")
                .withConf(".*")
                .withWrite(".*")
                .withRead(".*")
        );
        body.withVhosts(listbodyVhosts);
        body.withSecretKey("*****");
        body.withAccessKey("admin123");
        request.withBody(body);
        try {
            CreateUserResponse response = client.createUser(request);
            System.out.println(response.toString());
        } catch (ConnectionException e) {
            e.printStackTrace();
        } catch (RequestTimeoutException e) {
            e.printStackTrace();
        } catch (ServiceResponseException e) {
            e.printStackTrace();
        }
    }
}
```

```

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

```

Python

创建一个AMQP用户，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行配置、读、写。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = CreateUserRequest()
        request.instance_id = "{instance_id}"
        listVhostsbody = [
            AMQPUserPerm(
                vhost="default",
                conf=".*",
                write=".*",
                read=".*"
            )
        ]
        request.body = AMQPUser(
            vhosts=listVhostsbody,
            secret_key="*****",
            access_key="admin123"
        )
        response = client.create_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

```

Go

创建一个AMQP用户，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行配置、读、写。

```

package main

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

    request := &model.CreateUserRequest{}
    request.InstanceId = "{instance_id}"
    vhostVhosts:= "default"
    confVhosts:= ".*"
    writeVhosts:= ".*"
    readVhosts:= ".*"
    var listVhostsbody = []model.AmqpUserPerm{
        {
            Vhost: &vhostVhosts,
            Conf: &confVhosts,
            Write: &writeVhosts,
            Read: &readVhosts,
        },
    }
    secretKeyAmqpUser:= "*****"
    accessKeyAmqpUser:= "admin123"
    request.Body = &model.AmqpUser{
        Vhosts: &listVhostsbody,
        SecretKey: &secretKeyAmqpUser,
        AccessKey: &accessKeyAmqpUser,
    }
    response, err := client.CreateUser(request)
    if err == nil {
        fmt.Printf("%+v\n", response)
    } else {
        fmt.Println(err)
    }
}

```

更多

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

状态码

状态码	描述
200	创建用户成功。

错误码

请参见[错误码](#)。

5.8.2 查询用户列表

功能介绍

查询用户列表（仅AMQP版本支持）。

调用方法

请参见[如何调用API](#)。

URI

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

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。

表 5-88 Query 参数

参数	是否必选	参数类型	描述
offset	否	String	分页查询偏移量，表示从此偏移量开始查询，offset大于等于0，默认从0开始查询。
limit	否	String	分页查询单页数量，取值范围0~50，默认查询10条。

请求参数

无

响应参数

状态码：200

表 5-89 响应 Body 参数

参数	参数类型	描述
users	Array of AMQPUser objects	用户列表。
total	Integer	总用户个数。

表 5-90 AMQPUser

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： <ul style="list-style-type: none"> • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#%&*()-_+=\ [{]}:;'"<.>/?。 不能与名称或倒序的名称相同。
vhosts	Array of AMQPUserPerm objects	需要配置权限的Vhost，一个用户可以配置多个Vhost下的资源权限。

表 5-91 AMQPUserPerm

参数	参数类型	描述
vhost	String	需要配置权限的Vhost名称，一个用户可以配置多个Vhost下的资源权限。
conf	String	使用正则表达式匹配资源配置权限。例如，在输入框内输入“^janeway-.*”，则表示授权给该用户当前Vhost下，所有名称以“janeway-”开头的资源的配置权限。

参数	参数类型	描述
write	String	使用正则表达式匹配资源写权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的写权限。
read	String	使用正则表达式匹配资源读权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的读权限。

请求示例

查询用户列表。

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

响应示例

状态码：200

查询用户列表成功。

```
{
  "users": [ {
    "access_key": "admin123",
    "secret_key": "*****",
    "vhosts": [ {
      "vhost": "default",
      "conf": ".*",
      "write": ".*",
      "read": ".*"
    } ]
  } ],
  "total": 1
}
```

SDK 代码示例

SDK代码示例如下。

Java

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

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

public class ListUserSolution {

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

security risks. It is recommended that the AK and SK be stored in ciphertext in configuration files or environment variables and decrypted during use to ensure security.

// In this example, AK and SK are stored in environment variables for authentication. Before running this example, set environment variables CLOUD_SDK_AK and CLOUD_SDK_SK in the local environment

```
String ak = System.getenv("CLOUD_SDK_AK");
String sk = System.getenv("CLOUD_SDK_SK");
String projectId = "{project_id}";

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

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

Python

```
# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.valueOf("<YOUR REGION>")) \
        .build()

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

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

Go

```
package main

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

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

更多

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

状态码

状态码	描述
200	查询用户列表成功。

错误码

请参见[错误码](#)。

5.8.3 修改用户参数

功能介绍

修改用户参数（仅AMQP版本支持）。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-92 路径参数

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

请求参数

表 5-93 请求 Body 参数

参数	是否必选	参数类型	描述
access_key	否	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。
secret_key	否	String	密钥。 8-32个字符。 至少包含以下字符中的3种： <ul style="list-style-type: none"> ● 大写字母 ● 小写字母 ● 数字 ● 特殊字符`~!@#%&*()-_=+\ [{ } ; : " , < . > / ? 。 不能与名称或倒序的名称相同。

参数	是否必选	参数类型	描述
vhosts	否	Array of AMQPUserPerm objects	需要配置权限的Vhost，一个用户可以配置多个Vhost下的资源权限。

表 5-94 AMQPUserPerm

参数	是否必选	参数类型	描述
vhost	否	String	需要配置权限的Vhost名称，一个用户可以配置多个Vhost下的资源权限。
conf	否	String	使用正则表达式匹配资源配置权限。例如，在输入框内输入“^janeway-.*”，则表示授权给该用户当前Vhost下，所有名称以“janeway-”开头的资源的配置权限。
write	否	String	使用正则表达式匹配资源写权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的写权限。
read	否	String	使用正则表达式匹配资源读权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的读权限。

响应参数

状态码：200

表 5-95 响应 Body 参数

参数	参数类型	描述
access_key	String	用户名，只能英文字母开头，且由英文字母、数字、中划线、下划线组成，长度为7~64个字符。

参数	参数类型	描述
secret_key	String	密钥。 8-32个字符。 至少包含以下字符中的3种： <ul style="list-style-type: none"> • 大写字母 • 小写字母 • 数字 • 特殊字符`~!@#\$\$%^&*()-_+=\ [{]}:;'",<.>/?。 不能与名称或倒序的名称相同。
vhosts	Array of AMQPUserPerm objects	需要配置权限的Vhost，一个用户可以配置多个Vhost下的资源权限。

表 5-96 AMQPUserPerm

参数	参数类型	描述
vhost	String	需要配置权限的Vhost名称，一个用户可以配置多个Vhost下的资源权限。
conf	String	使用正则表达式匹配资源配置权限。例如，在输入框内输入“^janeway-.*”，则表示授权给该用户当前Vhost下，所有名称以“janeway-”开头的资源的配置权限。
write	String	使用正则表达式匹配资源写权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的写权限。
read	String	使用正则表达式匹配资源读权限。例如，在输入框内输入“.*”，则表示授权给该用户当前Vhost下，所有资源的读权限。

请求示例

修改用户参数，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行读、写、但仅允许对以janeway-开头的资源进行配置。

```
PUT https://{endpoint}/v2/{project_id}/instances/{instance_id}/users/{user_name}
{
  "access_key": "admin123",
  "secret_key": "*****",
  "vhosts": [ {
```

```
"vhost" : "default",
"conf" : "^janeway-.*",
"write" : ".*",
"read" : ".*"
}]
}
```

响应示例

状态码：200

修改用户参数成功。

```
{
  "access_key" : "admin123",
  "secret_key" : "*****",
  "vhosts" : [ {
    "vhost" : "default",
    "conf" : "^janeway-.*",
    "write" : ".*",
    "read" : ".*"
  } ]
}
```

SDK 代码示例

SDK代码示例如下。

Java

修改用户参数，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行读、写、但仅允许对以janeway-开头的资源进行配置。

```
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class UpdateUserSolution {

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

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

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

```

        .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
        .build();
    UpdateUserRequest request = new UpdateUserRequest();
    request.withInstanceId("{instance_id}");
    request.withUserName("{user_name}");
    AMQPUser body = new AMQPUser();
    List<AMQPUserPerm> listbodyVhosts = new ArrayList<>();
    listbodyVhosts.add(
        new AMQPUserPerm()
            .withVhost("default")
            .withConf("^janeway-.*")
            .withWrite(".*")
            .withRead(".*")
    );
    body.withVhosts(listbodyVhosts);
    body.withSecretKey("*****");
    body.withAccessKey("admin123");
    request.withBody(body);
    try {
        UpdateUserResponse response = client.updateUser(request);
        System.out.println(response.toString());
    } catch (ConnectionException e) {
        e.printStackTrace();
    } catch (RequestTimeoutException e) {
        e.printStackTrace();
    } catch (ServiceResponseException e) {
        e.printStackTrace();
        System.out.println(e.getHttpStatusCode());
        System.out.println(e.getRequestId());
        System.out.println(e.getErrorCode());
        System.out.println(e.getErrorMsg());
    }
}
}
}

```

Python

修改用户参数，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行读、写、但仅允许对以janeway-开头的资源进行配置。

```

# coding: utf-8

import os
from huaweicloudsdkcore.auth.credentials import BasicCredentials
from huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

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

```
listVhostsbody = [
    AMQPUserPerm(
        vhost="default",
        conf="^janeway-.*",
        write=".*",
        read=".*"
    )
]
request.body = AMQPUser(
    vhosts=listVhostsbody,
    secret_key="*****",
    access_key="admin123"
)
response = client.update_user(request)
print(response)
except exceptions.ClientRequestException as e:
    print(e.status_code)
    print(e.request_id)
    print(e.error_code)
    print(e.error_msg)
```

Go

修改用户参数，允许访问名称为default的Vhost，允许对此Vhost下所有资源进行读、写、但仅允许对以janeway-开头的资源进行配置。

```
package main

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

    request := &model.UpdateUserRequest{}
    request.InstanceId = "{instance_id}"
    request.UserName = "{user_name}"
    vhostVhosts := "default"
    confVhosts := "^janeway-.*"
    writeVhosts := ".*"
    readVhosts := ".*"
    var listVhostsbody = []model.AmqpUserPerm{
        {
            Vhost: &vhostVhosts,
            Conf: &confVhosts,
```

```

        Write: &writeVhosts,
        Read: &readVhosts,
    },
}
secretKeyAmqpUser:= "*****"
accessKeyAmqpUser:= "admin123"
request.Body = &model.AmqpUser{
    Vhosts: &listVhostsbody,
    SecretKey: &secretKeyAmqpUser,
    AccessKey: &accessKeyAmqpUser,
}
response, err := client.UpdateUser(request)
if err == nil {
    fmt.Printf("%+v\n", response)
} else {
    fmt.Println(err)
}
}
    
```

更多

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

状态码

状态码	描述
200	修改用户参数成功。

错误码

请参见[错误码](#)。

5.8.4 删除用户

功能介绍

删除用户（仅AMQP版本支持）。

调用方法

请参见[如何调用API](#)。

URI

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

表 5-97 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。
user_name	是	String	用户名。

请求参数

无

响应参数

无

请求示例

删除指定的用户。

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

响应示例

无

SDK 代码示例

SDK代码示例如下。

Java

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

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

public class DeleteUserSolution {

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

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



```

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
DeleteUserRequest request = new DeleteUserRequest();
request.withInstanceId("{instance_id}");
request.withUserName("{user_name}");
try {
    DeleteUserResponse response = client.deleteUser(request);
    System.out.println(response.toString());
} catch (ConnectionException e) {
    e.printStackTrace();
} catch (RequestTimeoutException e) {
    e.printStackTrace();
} catch (ServiceResponseException e) {
    e.printStackTrace();
    System.out.println(e.getStatusCode());
    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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = DeleteUserRequest()
        request.instance_id = "{instance_id}"
        request.user_name = "{user_name}"
        response = client.delete_user(request)
        print(response)
    except exceptions.ClientRequestException as e:
        print(e.status_code)
        print(e.request_id)
        print(e.error_code)
        print(e.error_msg)

```

Go

```

package main

import (
    "fmt"

```

```

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

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

```

更多

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

状态码

状态码	描述
204	删除成功。

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 5-98 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 5-99 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 5-100 响应 Body 参数

参数	参数类型	描述
task_count	String	任务数量。

参数	参数类型	描述
tasks	Array of tasks objects	任务列表。

表 5-101 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

查询任务列表成功。

```
{  
  "task_count": "1",  
  "tasks": [{  
    "id": "ff80808272dcc90f0172df1e490f41b0",  
    "name": "bindInstancePublicIp",  
    "user_name": "dms_test",  
    "user_id": "xxxxxxxx93ff484a828144c6xxxxxxxx",  
    "params": "{\"public_ip_id\":\"06a13350-4305-4338-9f0e-6b322bb1413d\", \"public_ip_address  
\": \"xx.xx.xx.xx\", \"enable_public_ip\": true}",  
    "status": "SUCCESS",  
    "created_at": "2020-06-23T03:00:03.471Z",  
    "updated_at": "2020-06-23T03:00:08.130Z"  
  }]  
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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	查询任务列表成功。

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 5-102 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	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_count	String	任务数量。
tasks	Array of tasks objects	任务列表。

表 5-104 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": "ff80808272dcc90f0172df1e490f41b0",
    "name": "bindInstancePublicIp",
    "user_name": "dms_test",
    "user_id": "xxxxxxxx93ff484a828144c6xxxxxxxx",
    "params": "{\"public_ip_id\":\"06a13350-4305-4338-9f0e-6b322bb1413d\", \"public_ip_address\": \"xx.xx.xx.xx\", \"enable_public_ip\": true}",
    "status": "SUCCESS",
    "created_at": "2020-06-23T03:00:03.471Z",
    "updated_at": "2020-06-23T03:00:08.130Z"
  } ]
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
    .with_credentials(credentials) \
    .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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 5-105 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	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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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.getStatusCode());
            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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion  
from huaweicloudsdkcore.exceptions import exceptions  
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \  
        .with_credentials(credentials) \  
        .with_region(RabbitMQRegion.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"  
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"  
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"  
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
    rabbitmq.RabbitMQClientBuilder().
        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}/rabbitmq/{instance_id}/tags/action

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。

请求参数

表 5-107 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): <ul style="list-style-type: none"> create (创建) delete (删除)
tags	否	Array of TagEntity objects	标签列表。

表 5-108 TagEntity

参数	是否必选	参数类型	描述
key	否	String	标签键。 <ul style="list-style-type: none"> 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。 <ul style="list-style-type: none"> 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}/rabbitmq/{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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

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

public class BatchCreateOrDeleteRabbitMqTagSolution {

    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);
```



```

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
    .build();
BatchCreateOrDeleteRabbitMqTagRequest request = new BatchCreateOrDeleteRabbitMqTagRequest();
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 {
    BatchCreateOrDeleteRabbitMqTagResponse response =
client.batchCreateOrDeleteRabbitMqTag(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.valueOf("<YOUR REGION>")) \
        .build()

    try:

```

```

request = BatchCreateOrDeleteRabbitMqTagRequest()
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_rabbit_mq_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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.BatchCreateOrDeleteRabbitMqTagRequest{
        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.BatchCreateOrDeleteRabbitMqTag(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}/rabbitmq/{instance_id}/tags

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 。

参数	是否必选	参数类型	描述
instance_id	是	String	实例ID。

请求参数

无

响应参数

状态码：200

表 5-110 响应 Body 参数

参数	参数类型	描述
tags	Array of TagEntity objects	标签列表

表 5-111 TagEntity

参数	参数类型	描述
key	String	标签键。 <ul style="list-style-type: none">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。 <ul style="list-style-type: none">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}/rabbitmq/{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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ShowRabbitMqTagsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowRabbitMqTagsRequest request = new ShowRabbitMqTagsRequest();
        request.withInstanceId("{instance_id}");
        try {
            ShowRabbitMqTagsResponse response = client.showRabbitMqTags(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.getStatusCode());
            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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.value_of("<YOUR REGION>")) \
        .build()

    try:
        request = ShowRabbitMqTagsRequest()
        request.instance_id = "{instance_id}"
        response = client.show_rabbit_mq_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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(  
    rabbitmq.RabbitMQClientBuilder().  
        WithRegion(region.ValueOf("<YOUR REGION>")).  
        WithCredential(auth).  
        Build())  
  
request := &model.ShowRabbitMqTagsRequest{}  
request.InstanceId = "{instance_id}"  
response, err := client.ShowRabbitMqTags(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}/rabbitmq/tags

表 5-112 路径参数

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

请求参数

无

响应参数

状态码: 200

表 5-113 响应 Body 参数

参数	参数类型	描述
tags	Array of TagMultyValueEntity objects	标签列表

表 5-114 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}/rabbitmq/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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.v2.model.*;

public class ShowRabbitMqProjectTagsSolution {

    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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.valueOf("<YOUR REGION>"))
            .build();
        ShowRabbitMqProjectTagsRequest request = new ShowRabbitMqProjectTagsRequest();
        try {
            ShowRabbitMqProjectTagsResponse response = client.showRabbitMqProjectTags(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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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")
    projectId = "{project_id}"

    credentials = BasicCredentials(ak, sk, projectId)

    client = RabbitMQClient.new_builder() \

```

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

try:
    request = ShowRabbitMqProjectTagsRequest()
    response = client.show_rabbit_mq_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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            WithRegion(region.ValueOf("<YOUR REGION>")).
            WithCredential(auth).
            Build())

    request := &model.ShowRabbitMqProjectTagsRequest{}
    response, err := client.ShowRabbitMqProjectTags(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 其他接口

5.11.1 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 5-115 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 5-116 MaintainWindowsEntity

参数	参数类型	描述
default	Boolean	是否为默认时间段。

参数	参数类型	描述
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

查询维护时间窗时间段成功。

```
{
  "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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

        RabbitMQClient client = RabbitMQClient.newBuilder()
            .withCredential(auth)
            .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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	查询维护时间窗时间段成功。

错误码

请参见[错误码](#)。

5.11.2 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 5-117 响应 Body 参数

参数	参数类型	描述
region_id	String	区域ID。
available_zones	Array of available_zones objects	可用区数组。

表 5-118 available_zones

参数	参数类型	描述
soldOut	Boolean	是否售罄。
id	String	可用区ID。
code	String	可用区编码。
name	String	可用区名称。
port	String	可用区端口号。
resource_availability	String	分区上是否还有可用资源。

参数	参数类型	描述
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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

    RabbitMQClient client = RabbitMQClient.newBuilder()
        .withCredential(auth)
        .withRegion(RabbitMQRegion.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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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](#)。

5.11.3 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 5-119 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 5-120 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 5-121 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 5-122 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。 <ul style="list-style-type: none"> 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 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 5-123 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 5-124 ListEnginePropertiesEntity

参数	参数类型	描述
step_length	String	节点增长步长。
max_queue_per_broker	String	每个Broker的最大队列。
max_connection_per_broker	String	每个Broker的最大连接数。

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/rabbitmq/products

响应示例

状态码：200

查询产品规格列表成功。

查询产品规格列表成功。

```
{
  "engine": "rabbitmq",
  "versions": [ "3.8.35" ],
  "products": [ {
    "type": "single",
    "product_id": "c6.2u4g.single",
    "ecs_flavor_id": "c6.large.2",
    "billing_code": "dms.platinum.c6",
    "arch_types": [ "X86" ],
    "charging_mode": [ "monthly", "hourly" ],
    "ios": [ {
      "io_spec": "dms.physical.storage.ultra.v2",
      "type": "evs",
      "available_zones": [ "xxx" ],
      "unavailable_zones": [ "xxx" ]
    }, {
      "io_spec": "dms.physical.storage.high.v2",
      "type": "evs",
      "available_zones": [ "xxx" ],
      "unavailable_zones": [ "xxx" ]
    } ],
    "support_features": [ ],
    "properties": {
```

```

    "max_connection_per_broker" : "2000",
    "max_broker" : "1",
    "max_queue_per_broker" : "100",
    "max_storage_per_node" : "30000",
    "min_broker" : "1",
    "step_length" : "0",
    "min_storage_per_node" : "200",
    "product_alias" : "rabbitmq.2u4g.single"
  }
}, {
  "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" ],
    "unavailable_zones" : [ "xxx" ]
  }, {
    "io_spec" : "dms.physical.storage.ultra.v2",
    "type" : "evs",
    "available_zones" : [ "xxx" ],
    "unavailable_zones" : [ "xxx" ]
  } ],
  "support_features" : [ ],
  "properties" : {
    "max_connection_per_broker" : "4500",
    "max_broker" : "7",
    "max_queue_per_broker" : "400",
    "max_storage_per_node" : "30000",
    "min_broker" : "3",
    "step_length" : "2",
    "min_storage_per_node" : "100",
    "product_alias" : "rabbitmq.4u8g.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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.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.getStatusCode());
    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 huaweicloudsdrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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](#)。

5.11.4 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 5-125 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

无

响应参数

状态码：200

表 5-126 响应 Body 参数

参数	参数类型	描述
dimensions	Array of dimensions objects	监控维度。
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	Queue信息。
vhosts	Array of vhosts objects	Vhost信息

参数	参数类型	描述
exchanges	Array of exchanges objects	exchange信息
groups	Array of groups objects	消费组信息。

表 5-127 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 5-128 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 5-129 instance_ids

参数	参数类型	描述
name	String	实例ID。

表 5-130 nodes

参数	参数类型	描述
name	String	节点名称。
available_zone	String	可用区。

表 5-131 queues

参数	参数类型	描述
name	String	Queue名称。
vhost	String	对应的Vhost。

表 5-132 vhosts

参数	参数类型	描述
name	String	Vhost名称。

表 5-133 exchanges

参数	参数类型	描述
name	String	Exchange名称。
vhost	String	对应的Vhost。

表 5-134 groups

参数	参数类型	描述
name	String	消费组名称。

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": "rabbitmq_instance_id",
    "metrics": [ "connections", "channels", "queues", "consumers", "messages_ready",
"messages_unacknowledged", "publish", "deliver", "deliver_no_ack", "deliver_get", "instance_bytes_in_rate",
"instance_bytes_out_rate", "instance_disk_usage" ],
    "key_name": [ "instance_ids" ],
    "dim_router": [ "rabbitmq_instance_id" ]
  }, {
    "name": "rabbitmq_node",
    "metrics": [ "fd_used", "socket_used", "proc_used", "mem_used", "disk_free", "rabbitmq_alive",
```

```
"rabbitmq_disk_usage", "rabbitmq_cpu_usage", "rabbitmq_cpu_core_load", "rabbitmq_memory_usage",
"rabbitmq_disk_read_await", "rabbitmq_disk_write_await", "rabbitmq_node_bytes_in_rate",
"rabbitmq_node_bytes_out_rate", "rabbitmq_node_queues", "rabbitmq_memory_high_watermark",
"rabbitmq_disk_insufficient" ],
  "key_name": [ "nodes" ],
  "dim_router": [ "rabbitmq_instance_id", "rabbitmq_node" ]
}, {
  "name": "rabbitmq_queue",
  "metrics": [ "queue_messages_unacknowledged", "queue_messages_ready" ],
  "key_name": [ "queues" ],
  "dim_router": [ "rabbitmq_instance_id", "rabbitmq_queue" ]
}],
"instance_ids": [ {
  "name": "0e16280d-7451-4f5b-80fa-f210372ce657"
}],
"nodes": [ {
  "name": "dms-vm-0e16280d-rabbitmq-0",
  "available_zone": "cn-north-7c"
}, {
  "name": "dms-vm-0e16280d-rabbitmq-1",
  "available_zone": "cn-north-7c"
}, {
  "name": "dms-vm-0e16280d-rabbitmq-2",
  "available_zone": "cn-north-7c"
}],
"queues": [ {
  "name": "Vhost-17130843_Queue-21084756",
  "vhost": "default"
}],
"vhosts": [ {
  "name": "default"
}],
"exchanges": [ {
  "name": "direct_exchange",
  "vhost": "default"
}],
"groups": [ ]
}
```

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.rabbitmq.v2.region.RabbitMQRegion;
import com.huaweicloud.sdk.rabbitmq.v2.*;
import com.huaweicloud.sdk.rabbitmq.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);

RabbitMQClient client = RabbitMQClient.newBuilder()
    .withCredential(auth)
    .withRegion(RabbitMQRegion.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.getStatusCode());
    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 huaweicloudsdkrabbitmq.v2.region.rabbitmq_region import RabbitMQRegion
from huaweicloudsdkcore.exceptions import exceptions
from huaweicloudsdkrabbitmq.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 = RabbitMQClient.new_builder() \
        .with_credentials(credentials) \
        .with_region(RabbitMQRegion.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"
    rabbitmq "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2"
    "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/v2/model"
    region "github.com/huaweicloud/huaweicloud-sdk-go-v3/services/rabbitmq/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 := rabbitmq.NewRabbitMQClient(
        rabbitmq.RabbitMQClientBuilder().
            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](#)。

5.11.5 查询 RabbitMQ 产品规格核数

功能介绍

查询RabbitMQ产品规格核数。

调用方法

请参见[如何调用API](#)。

URI

GET /v2/rabbitmq/products/cores

表 5-135 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 5-136 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/rabbitmq/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DMS for RabbitMQ实例进行精细的权限管理，您可以使用统一身份认证服务（Identity and Access Management，简称IAM），如果华为账号已经能满足您的要求，不需要创建独立的IAM用户，您可以跳过本章节，不影响您使用DMS for RabbitMQ实例的其它功能。

默认情况下，新建的IAM用户没有任何权限，您需要将其加入用户组，并给用户组授予策略或角色，才能使用户组中的用户获得相应的权限，这一过程称为授权。授权后，用户就可以基于已有权限对云服务进行操作。

权限根据授权的精细程度，分为**角色**和**策略**。角色以服务为粒度，是IAM最初提供的一种根据用户的工作职能定义权限的粗粒度授权机制。策略以API接口为粒度进行权限拆分，授权更加精细，可以精确到某个操作、资源和条件，能够满足企业对权限最小化的安全管控要求。

说明

如果您要允许或是禁止某个接口的操作权限，请使用策略。

账号具备所有接口的调用权限，如果使用账号下的IAM用户发起API请求时，该IAM用户必须具备调用该接口所需的权限，否则，API请求将调用失败。每个接口所需要的权限，与各个接口所对应的授权项相对应，只有发起请求的用户被授予授权项所对应的策略，该用户才能成功调用该接口。例如，用户要调用接口来创建RabbitMQ实例，那么这个IAM用户被授予的策略中必须包含允许“dms:instance:create”的授权项，该接口才能调用成功。

支持的授权项

策略包含系统策略和自定义策略，如果系统策略不满足授权要求，管理员可以创建自定义策略，并通过给用户组授予自定义策略来进行精细的访问控制。策略支持的操作与API相对应，授权项列表说明如下：

- 权限：允许或拒绝某项操作。
- 对应API接口：自定义策略实际调用的API接口。
- 授权项：自定义策略中支持的Action，在自定义策略中的Action中写入授权项，可以实现授权项对应的权限功能。
- IAM项目（Project）/企业项目（Enterprise Project）：自定义策略的授权范围，包括IAM项目与企业项目。授权范围如果同时支持IAM项目和企业项目，表示此授权项对应的自定义策略，可以在IAM和企业管理两个服务中给用户组授权并生

效。如果仅支持IAM项目，不支持企业项目，表示仅能在IAM中给用户组授权并生效，如果在企业管理中授权，则该自定义策略不生效。关于IAM项目与企业项目的区别，详情请参见：[IAM与企业管理的区别](#)。

DMS for RabbitMQ管理控制操作相关的企业项目细粒度权限授权项，明细如下表所示。用户调用如下API时，需要获取对应的权限。权限获取请参考[统一身份认证服务 \(IAM\)](#)的帮助指导。

表 6-1 DMS for RabbitMQ 授权项明细

权限	对应API接口	授权项	IAM项目 (Project)	企业项目 (Enterprise Project)
创建实例	POST /v2/{project_id}/instances	dms:instance:create	√	√
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	√	√
查询指定实例	GET /v2/{project_id}/instances/{instance_id}	dms:instance:get	√	√
批量删除实例	POST /v2/{project_id}/instances/action	dms:instance:delete	√	√
查询所有实例列表	GET /v2/{project_id}/instances	dms:instance:list	√	√
查询插件列表	GET /v2/{project_id}/instances/{instance_id}/rabbitmq/plugins	dms:instance:list	√	√
开启或关闭插件	PUT /v2/{project_id}/instances/{instance_id}/rabbitmq/plugins	dms:instance:modify	√	√

权限	对应API接口	授权项	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	√	√
实例规格变更	POST /v2/{project_id}/instances/{instance_id}/extend	dms:instance:scale	√	√
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	GET /v2/{project_id}/instances/{instance_id}/tasks/{task_id}	dms:instance:deleteBackgroundTask	√	√
批量添加或删除实例标签	POST /v2/{project_id}/rabbitmq/{instance_id}/tags/action	dms:instance:modify	√	√
查询实例标签	GET /v2/{project_id}/rabbitmq/{instance_id}/tags	dms:instance:get	√	√
查询项目标签	GET /v2/{project_id}/rabbitmq/tags	dms:instance:get	√	√
开启关闭公网访问	PUT /v2/{project_id}/instances/{instance_id}	dms:instance:modify	√	√

7 历史 API

7.1 API V1

7.1.1 实例管理类接口

7.1.1.1 创建实例

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 \"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：rabbitmq。
engine_version	String	否	消息引擎的版本。
storage_space	Integer	是	消息存储空间，单位GB。 <ul style="list-style-type: none"> • 单机RabbitMQ实例的存储空间的取值范围100GB~90000GB。 • 集群RabbitMQ实例的存储空间的取值范围为100GB*节点数~90000GB、200GB*节点数~90000GB、300GB*节点数~90000GB。
access_user	String	是	认证用户名，只能由英文字母、数字、中划线组成，长度为4~64的字符。
password	String	是	实例的认证密码。 复杂度要求： <ul style="list-style-type: none"> • 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两种组合： <ul style="list-style-type: none"> - 小写字母 - 大写字母 - 数字 - 特殊字符包括 (`~!@#\$\$%^&*()-_+=\ [{}]:\"<,.>/?)
vpc_id	String	是	租户VPC ID。
security_group_id	String	是	租户安全组ID。
subnet_id	String	是	子网ID。
available_zones	Array	是	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，详情请参考 查询可用区信息 查询得到。

参数	类型	是否必选	说明
product_id	String	是	产品标识。 详情请参考 查询产品规格列表 。
maintain_begin	String	否	维护时间窗开始时间，格式为HH:mm。 <ul style="list-style-type: none"> 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、02:00、06:00、10:00、14:00和18:00。 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。
maintain_end	String	否	维护时间窗结束时间，格式为HH:mm。 <ul style="list-style-type: none"> 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时，结束时间为02:00。 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。
enable_publicip	Boolean	否	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> true: 开启 false: 不开启
publicip_id	String	否	RabbitMQ实例绑定的弹性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。 <ul style="list-style-type: none"> true: 打开SSL加密访问。 false: 不打开SSL加密访问。
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： <ul style="list-style-type: none"> dms.physical.storage.normal: dms.physical.storage.high dms.physical.storage.ultra
enterprise_project_id	String	否	企业项目ID。

RabbitMQ实例的请求示例

```
{
  "name": "rabbitmq-demo",
  "description": "",
  "engine": "RabbitMQ",
  "engine_version": "3.x.x",
  "storage_space": 100,
  "access_user": "*****",
  "password": "*****",
  "vpc_id": "1e93f86e-13af-46c8-97d6-d40fa62b76c2",
  "security_group_id": "0aaa0033-bf7f-4c41-a6c2-18cd04cad2c8",
  "subnet_id": "b5fa806c-35e7-4299-b659-b39398dd4718",
  "available_zones": ["d573142f24894ef3bd3664de068b44b0"],
  "product_id": "00300-30109-0--0",
  "maintain_begin": "22:00",
  "maintain_end": "02:00",
  "ssl_enable": false,
  "enable_publicip": false,
  "publicip_id": "",
  "enterprise_project_id": "0",
  "storage_spec_code": "dms.physical.storage.ultra"
}
```

响应消息

响应参数

参数说明见[表7-3](#)。

表 7-3 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7-4](#)所示，其他响应见[状态码](#)。

表 7-4 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7-5](#)。

表 7-5 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

无。

响应消息

响应参数

参数说明见[表7-6](#)。

表 7-6 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。 <ul style="list-style-type: none">• RabbitMQ实例单机返回vm规格。• RabbitMQ实例集群返回vm规格和节点数。
storage_space	Integer	消息存储空间，单位：GB。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。

参数	类型	说明
connect_addresses	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 。
description	String	实例描述。
instance_id	String	实例ID。
resource_spec_code	String	资源规格标识。 <ul style="list-style-type: none"> dms.instance.rabbitmq.single.c3.2u4g: RabbitMQ单机, vm规格2u4g dms.instance.rabbitmq.single.c3.4u8g: RabbitMQ单机, vm规格4u8g dms.instance.rabbitmq.single.c3.8u16g: RabbitMQ单机, vm规格8u16g dms.instance.rabbitmq.single.c3.16u32g: RabbitMQ单机, vm规格16u32g dms.instance.rabbitmq.cluster.c3.4u8g.3: RabbitMQ集群, vm规格4u8g, 3个节点 dms.instance.rabbitmq.cluster.c3.4u8g.5: RabbitMQ集群, vm规格4u8g, 5个节点 dms.instance.rabbitmq.cluster.c3.4u8g.7: RabbitMQ集群, vm规格4u8g, 7个节点 dms.instance.rabbitmq.cluster.c3.8u16g.3: RabbitMQ集群, vm规格8u16g, 3个节点 dms.instance.rabbitmq.cluster.c3.8u16g.5: RabbitMQ集群, vm规格8u16g, 5个节点 dms.instance.rabbitmq.cluster.c3.8u16g.7: RabbitMQ集群, vm规格8u16g, 7个节点 dms.instance.rabbitmq.cluster.c3.16u32g.3: RabbitMQ集群, vm规格16u32g, 3个节点 dms.instance.rabbitmq.cluster.c3.16u32g.5: RabbitMQ集群, vm规格16u32g, 5个节点 dms.instance.rabbitmq.cluster.c3.16u32g.7: RabbitMQ集群, vm规格16u32g, 7个节点
type	String	实例类型。 <ul style="list-style-type: none"> 单机: single 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。

参数	类型	说明
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。
error_code	String	实例创建失败或状态异常时的错误码，错误码说明见表7-7。
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，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。
maintain_end	String	维护时间窗结束时间，格式为HH:mm。
enable_publicip	Boolean	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
publicip_addresses	String	RabbitMQ实例绑定的弹性IP地址。 如果未开启公网访问功能，该字段值为null。
publicip_id	String	RabbitMQ实例绑定的弹性IP地址的ID。 如果未开启公网访问功能，该字段值为null。
management_connect_address	String	RabbitMQ实例的管理地址。
ssl_enable	Boolean	是否开启安全认证。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启

参数	类型	说明
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。 <ul style="list-style-type: none"> • 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	String	实例扩容磁盘次数, 如果超过20次则无法扩容磁盘。

表 7-7 错误码说明

错误码	说明
public.00.0001	内部服务错误。
public.00.0002	内部服务错误。
public.00.0003	内部服务错误。
public.00.0004	VPC创建失败。
public.00.0005	安全组创建失败。
public.00.0006	子网创建失败。
public.00.0007	子网状态异常。
public.00.0008	创建ECS失败。
public.00.0009	创建ECS失败。
public.00.0010	创建ECS失败。
public.00.0011	ECS绑定网卡失败。
public.00.0013	ECS启动失败。
public.00.0014	ECS启动失败。
public.00.0015	ECS停止失败。
public.00.0018	创建ECS失败, ECS资源配额不足。
public.00.0024	实例部署异常。
public.00.0025	实例部分节点故障。
public.00.0042	无法连接实例。

响应示例

```
{
  "name": "dms-a11e",
  "engine": "rabbitmq",
  "engine_version": "3.x.x",
  "specification": "2vCPUs 4GB",
  "storage_space": 100,
  "used_storage_space": 50,
  "connect_address": "192.168.3.100",
  "port": 5672,
  "status": "RUNNING",
  "description": "Create a instance",
  "instance_id": "68d5745e-6af2-40e4-945d-fe449be00148",
  "resource_spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
  "type": "single",
  "charging_mode": 1,
  "vpc_id": "27d99e17-42f2-4751-818f-5c8c6c03ff15",
  "vpc_name": "vpc_4944a40e-ac57-4f08-9d38-9786e2759458_192",
  "created_at": "1526367063931",
  "error_code": null,
  "product_id": "00300-30109-0--0",
  "security_group_id": "60ea2db8-1a51-4ab6-9e11-65b418c24583",
  "security_group_name": "sg_6379_4944a40e-ac57-4f08-9d38-9786e2759458",
  "subnet_id": "ec2f34b9-20eb-4872-85bd-bea9fc943128",
  "subnet_name": "subnet_az_7f336767-10ec-48a5-9ae8-9cacde119318",
  "subnet_cidr": "192.168.0.0/24",
  "available_zones": ["1d7b939b382c4c3bb3481a8ca10da785"],
  "user_id": "6d0977e4c9b74ae7b5a083a8d0d8fafa",
  "user_name": "aabb02",
  "access_user": "user",
  "order_id": "XXXXXXXXX",
  "maintain_begin": "22:00",
  "maintain_end": "02:00",
  "enable_publicip": "true",
  "publicip_id": "b7940732-11ef-459b-acab-cab0d26c74a3",
  "publicip_address": "192.168.10.5",
  "ssl_enable": false,
  "management_connect_address": "http://192.168.0.177:9999"
}
```

状态码

操作成功的状态码如[表7-8](#)所示，其他响应见[状态码](#)。

表 7-8 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

修改实例的名称和描述信息。

URI

PUT /v1.0/{project_id}/instances/{instance_id}

表 7-9 参数说明

参数	类型	必选	备注
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7-10。

表 7-10 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\"。
maintain_begin	String	否	维护时间窗开始时间，格式为HH:mm:ss。 <ul style="list-style-type: none"> 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。

参数	类型	必选	说明
maintain_end	String	否	维护时间窗结束时间，格式为HH:mm:ss。 <ul style="list-style-type: none"> 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。
enable_publicip	Boolean	否	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> true: 开启 false: 不开启
publicip_id	String	否	RabbitMQ实例绑定的弹性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。
enterprise_project_id	String	否	企业项目ID。

请求示例

示例1：

```
{
  "name": "dms002",
  "description": "instance description"
}
```

示例2

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

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7-11](#)所示，其他响应见[状态码](#)。

表 7-11 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7-12](#)。

表 7-12 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

无。

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7-13](#)所示，其他响应见[状态码](#)。

表 7-13 状态码

状态码	描述
204	删除实例成功。

7.1.1.5 批量删除实例

说明

当前页面API为历史版本API，未来可能停止维护。请使用[批量删除实例](#)。

功能介绍

批量删除实例。

实例删除后，实例中原有的数据将被删除，且没有备份，请谨慎操作。

URI

POST /v1.0/{project_id}/instances/action

参数说明见[表7-14](#)。

表 7-14 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7-15](#)。

表 7-15 参数说明

参数	类型	必选	说明
action	String	是	对实例的操作：delete
instances	Array	是	实例的ID列表。

参数	类型	必选	说明
allFailure	String	否	是否批量删除创建失败的实例。 当参数值为“true”时，删除租户所有创建失败的实例，此时请求参数 instances 可为空。

请求示例

批量删除实例

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

删除所有创建失败的实例

```
{
  "action": "delete",
  "allFailure": "true"
}
```

响应消息

响应参数

当参数 action 为 delete，allFailure 值为 true 时，响应返回为空表示删除成功。参数说明见表 7-16。

表 7-16 参数说明

参数	类型	说明
results	Array	修改实例的结果。

表 7-17 results 参数说明

参数	类型	说明
instance	String	实例 ID。
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7-18](#)所示，其他响应见[状态码](#)。

表 7-18 状态码

状态码	描述
200	删除实例成功。

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7-19](#)。

表 7-19 参数说明

参数	类型	必选	说明
project_id	String	是	项目ID。
engine	String	否	rabbitmq，参数缺失查询所有实例。
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 。

参数	类型	必选	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。

示例

```
GET /v1.0/bd6b78e2ff9e4e47bc260803ddcc7a21/instances?
start=1&limit=10&name=&status=&id=&includeFailure=true&exactMatchName=false
```

请求消息

请求参数

无。

请求示例

无。

响应消息

响应参数

参数说明见表7-20。

表 7-20 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 7-21 instance 参数说明

参数	类型	说明
name	String	实例名称。
engine	String	引擎。

参数	类型	说明
engine_version	String	版本。
specification	String	实例规格。 <ul style="list-style-type: none"> • RabbitMQ实例单机返回vm规格。 • RabbitMQ实例集群返回vm规格和节点数。
storage_space	Integer	消息存储空间，单位：GB。
used_storage_space	Integer	已使用的消息存储空间，单位：GB。
connect_addresses	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 。
description	String	实例描述。
instance_id	String	实例ID。

参数	类型	说明
resource_spec_code	String	资源规格标识。 <ul style="list-style-type: none"> dms.instance.rabbitmq.single.c3.2u4g: RabbitMQ单机, vm规格2u4g dms.instance.rabbitmq.single.c3.4u8g: RabbitMQ单机, vm规格4u8g dms.instance.rabbitmq.single.c3.8u16g: RabbitMQ单机, vm规格8u16g dms.instance.rabbitmq.single.c3.16u32g: RabbitMQ单机, vm规格16u32g dms.instance.rabbitmq.cluster.c3.4u8g.3: RabbitMQ集群, vm规格4u8g, 3个节点 dms.instance.rabbitmq.cluster.c3.4u8g.5: RabbitMQ集群, vm规格4u8g, 5个节点 dms.instance.rabbitmq.cluster.c3.4u8g.7: RabbitMQ集群, vm规格4u8g, 7个节点 dms.instance.rabbitmq.cluster.c3.8u16g.3: RabbitMQ集群, vm规格8u16g, 3个节点 dms.instance.rabbitmq.cluster.c3.8u16g.5: RabbitMQ集群, vm规格8u16g, 5个节点 dms.instance.rabbitmq.cluster.c3.8u16g.7: RabbitMQ集群, vm规格8u16g, 7个节点 dms.instance.rabbitmq.cluster.c3.16u32g.3: RabbitMQ集群, vm规格16u32g, 3个节点 dms.instance.rabbitmq.cluster.c3.16u32g.5: RabbitMQ集群, vm规格16u32g, 5个节点 dms.instance.rabbitmq.cluster.c3.16u32g.7: RabbitMQ集群, vm规格16u32g, 7个节点
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。
error_code	String	实例创建失败或状态异常时的错误码, 错误码说明见 表7-7 。
user_id	String	用户id。
user_name	String	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。

参数	类型	说明
maintain_begin	String	维护时间窗开始时间，格式为HH:mm。
maintain_end	String	维护时间窗结束时间，格式为HH:mm。
enable_publicip	Boolean	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
publicip_addresses	String	RabbitMQ实例绑定的弹性IP地址。 如果未开启公网访问功能，该字段值为null。
publicip_id	String	RabbitMQ实例绑定的弹性IP地址的ID。 如果未开启公网访问功能，该字段值为null。
management_connect_address	String	RabbitMQ实例的管理地址。
ssl_enable	Boolean	是否开启安全认证。 <ul style="list-style-type: none"> • 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。 <ul style="list-style-type: none"> • 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	String	实例扩容磁盘次数，如果超过20次则无法扩容磁盘。

响应示例

```
{
  "instances": [
    {
      "name": "rabbitmq-lxy001",
      "engine": "rabbitmq",
      "port": 5672,
      "status": "RUNNING",
      "type": "single",
      "specification": "2vCPUs 4GB",
      "engine_version": "3.x.x",
      "connect_address": "192.168.255.237",
      "instance_id": "595926bf-a648-47d8-91bc-461956794c2b",
      "resource_spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
      "charging_mode": 1,
      "vpc_id": "1a28dcc5-c90d-421c-82bb-783f30f5b40a",
      "vpc_name": "vpc-y00292973",
      "created_at": "1562583302800",
      "product_id": "00300-30109-0--0",
      "security_group_id": "0cc8fdb7-872a-49da-a062-88ccc39463b5",
      "security_group_name": "sg-65eb-nw-test",
    }
  ]
}
```

```

"subnet_id": "ebba7994-260d-42ab-bce1-39a08b365dc8",
"available_zones": [
  "d573142f24894ef3bd3664de068b44b0"
],
"user_id": "50a4156d334a4a82b8745dc730dc1e00",
"user_name": "test",
"access_user": "test-Info",
"maintain_begin": "02:00:00",
"maintain_end": "06:00:00",
"storage_space": 88,
"total_storage_space": 100,
"used_storage_space": 4,
"enable_publicip": false,
"ssl_enable": false,
"management_connect_address": "http://192.168.255.237:15672",
"storage_resource_id": "34825335-61cb-4ee0-949e-24b08170edb2",
"storage_spec_code": "dms.physical.storage.ultra",
"service_type": "advanced",
"storage_type": "hec",
"enterprise_project_id": "0",
"is_logical_volume": false,
"extend_times": 0,
"ipv6_enable": false,
"ipv6_connect_addresses": [],
"connector_enable": false,
"connector_id": "",
"rest_enable": false,
"rest_connect_address": "",
"public_boundwidth": 0,
"message_query_inst_enable": true,
"vpc_client_plain": false,
"support_features":
"feature.physerver.kafka.topic.accesspolicy,message_trace_enable,feature.physerver.kafka.pulbic.dynamic,feat
ure.physerver.kafka.user.manager",
  "trace_enable": false
}
],
"instance_num": 1
}

```

状态码

操作成功的状态码如[表7-22](#)所示，其他响应见[状态码](#)。

表 7-22 状态码

状态码	描述
200	查询所有实例列表成功。

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

无。

响应消息

响应参数

参数说明见[表7-23](#)、[表7-24](#)。

表 7-23 参数说明

参数	类型	说明
regionId	String	区域ID。
available_zones	Array	可用区数组，具体请参考 表7-24 。

表 7-24 available_zones 参数说明

参数	类型	说明
id	String	可用区ID。
code	String	可用区编码。
name	String	可用区名称。
port	String	可用区端口号。
resource_availability	String	分区上是否还有可用资源。 <ul style="list-style-type: none">• true: 还有资源。• false: 资源已售罄。

响应示例

```
{  
  regionId: "XXXXXX",  
}
```



```
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7-25](#)所示，其他响应见[状态码](#)。

表 7-25 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。

URI

GET /v1.0/products?engine={engine}

参数说明见[表7-26](#)。

表 7-26 参数说明

参数	类型	必选	说明
engine	String	否	消息引擎的类型。

请求消息

请求参数

无。

请求示例

无。

响应消息

响应参数

Hourly或者Monthly的参数说明见[表7-27](#)。

表 7-27 参数说明

参数	类型	备注
name	String	消息引擎的名称。
version	String	消息引擎的版本。
values	Array	产品规格列表。

表 7-28 values 参数说明

参数	类型	备注
detail	Array	规格详情。
name	String	实例类型，单机或者集群。

表 7-29 RabbitMQ 单机实例的 detail 参数说明

参数	类型	备注
storage	String	消息存储空间。
io	Array	IO信息。
vm_specification	String	虚拟机规格。
product_id	String	产品ID。
spec_code	String	规格ID。

表 7-30 RabbitMQ 集群实例的 detail 参数说明

参数	类型	备注
vm_specification	String	虚拟机规格。
product_info	Array	产品信息。

表 7-31 product_info 参数说明

参数	类型	备注
storage	String	消息存储空间。
io	Array	IO信息。
node_num	Integer	集群的节点个数。
product_id	String	产品ID。
spec_code	String	规格ID。

表 7-32 io 参数说明

参数	类型	备注
io_type	String	IO类型。
storage_spec_code	String	IO规格。

响应示例

RabbitMQ规格列表：

```
{
  "Hourly": [
    {
      "name": "RabbitMQ",
      "version": "3.x.x",
      "values": [
        {
          "detail": [
            {
              "storage": "100",
              "io": [
                {
                  "io_type": "normal",
                  "storage_spec_code": "dms.physical.storage.normal"
                },
                {
                  "io_type": "high",
                  "storage_spec_code": "dms.physical.storage.high"
                },
                {
                  "io_type": "ultra",
                  "storage_spec_code": "dms.physical.storage.ultra"
                }
              ]
            },
            {
              "vm_specification": "2vCPUs 4GB",
              "product_id": "00300-30109-0--0",
              "spec_code": "dms.instance.rabbitmq.single.c3.2u4g"
            }
          ]
        },
        {
          "storage": "100",
          "io": [
            {
              "io_type": "normal",
              "storage_spec_code": "dms.physical.storage.normal"
            }
          ]
        }
      ]
    }
  ]
}
```

```
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "4vCPUs 8GB",
  "product_id": "00300-30111-0--0",
  "spec_code": "dms.instance.rabbitmq.single.c3.4u8g"
},
{
  "storage": "100",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "8vCPUs 16GB",
  "product_id": "00300-30113-0--0",
  "spec_code": "dms.instance.rabbitmq.single.c3.8u16g"
},
{
  "storage": "100",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "16vCPUs 32GB",
  "product_id": "00300-30115-0--0",
  "spec_code": "dms.instance.rabbitmq.single.c3.16u32g"
}
],
"name": "single"
},
{
  "detail": [
    {
      "vm_specification": "4vCPUs 8GB",
      "product_info": [
        {
          "storage": "300",
          "io": [
            {
              "io_type": "normal",
              "storage_spec_code": "dms.physical.storage.normal"
            }
          ]
        }
      ]
    }
  ]
}
```

```
{
  {
    "io_type": "high",
    "storage_spec_code": "dms.physical.storage.high"
  },
  {
    "io_type": "ultra",
    "storage_spec_code": "dms.physical.storage.ultra"
  }
],
"node_num": "3",
"product_id": "00300-30209-0--0",
"spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.3"
},
{
  "storage": "500",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "node_num": "5",
  "product_id": "00300-30211-0--0",
  "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.5"
},
{
  "storage": "700",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "node_num": "7",
  "product_id": "00300-30213-0--0",
  "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.7"
}
]
},
{
  "vm_specification": "8vCPUs 16GB",
  "product_info": [
    {
      "storage": "300",
      "io": [
        {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal"
        },
        {
          "io_type": "high",
          "storage_spec_code": "dms.physical.storage.high"
        }
      ],
    }
  ],
}
```

```
{
  "io_type": "ultra",
  "storage_spec_code": "dms.physical.storage.ultra"
},
],
"node_num": "3",
"product_id": "00300-30215-0--0",
"spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.3"
},
{
  "storage": "500",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "node_num": "5",
  "product_id": "00300-30217-0--0",
  "spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.5"
},
{
  "storage": "700",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "node_num": "7",
  "product_id": "00300-30219-0--0",
  "spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.7"
}
],
},
{
  "vm_specification": "16vCPUs 32GB",
  "product_info": [
    {
      "storage": "300",
      "io": [
        {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal"
        },
        {
          "io_type": "high",
          "storage_spec_code": "dms.physical.storage.high"
        },
        {
          "io_type": "ultra",
          "storage_spec_code": "dms.physical.storage.ultra"
        }
      ]
    }
  ]
}
```

```
    ],  
    "node_num": "3",  
    "product_id": "00300-30221-0--0",  
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.3"  
  },  
  {  
    "storage": "500",  
    "io": [  
      {  
        "io_type": "normal",  
        "storage_spec_code": "dms.physical.storage.normal"  
      },  
      {  
        "io_type": "high",  
        "storage_spec_code": "dms.physical.storage.high"  
      },  
      {  
        "io_type": "ultra",  
        "storage_spec_code": "dms.physical.storage.ultra"  
      }  
    ],  
    "node_num": "5",  
    "product_id": "00300-30223-0--0",  
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.5"  
  },  
  {  
    "storage": "700",  
    "io": [  
      {  
        "io_type": "normal",  
        "storage_spec_code": "dms.physical.storage.normal"  
      },  
      {  
        "io_type": "high",  
        "storage_spec_code": "dms.physical.storage.high"  
      },  
      {  
        "io_type": "ultra",  
        "storage_spec_code": "dms.physical.storage.ultra"  
      }  
    ],  
    "node_num": "7",  
    "product_id": "00300-30225-0--0",  
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.7"  
  }  
] ]  
},  
"name": "cluster"  
}  
]  
},  
"Monthly": [  
  {  
    "name": "RabbitMQ",  
    "version": "3.x.x",  
    "values": [  
      {  
        "detail": [  
          {  
            "storage": "100",  
            "io": [  
              {  
                "io_type": "normal",  
                "storage_spec_code": "dms.physical.storage.normal"  
              },  
              {  
                "io_type": "high",
```

```
    "storage_spec_code": "dms.physical.storage.high"
  },
  {
    "io_type": "ultra",
    "storage_spec_code": "dms.physical.storage.ultra"
  }
],
"vm_specification": "2vCPUs 4GB",
"product_id": "00300-30110-0--0",
"spec_code": "dms.instance.rabbitmq.single.c3.2u4g"
},
{
  "storage": "100",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "4vCPUs 8GB",
  "product_id": "00300-30112-0--0",
  "spec_code": "dms.instance.rabbitmq.single.c3.4u8g"
},
{
  "storage": "100",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "8vCPUs 16GB",
  "product_id": "00300-30114-0--0",
  "spec_code": "dms.instance.rabbitmq.single.c3.8u16g"
},
{
  "storage": "100",
  "io": [
    {
      "io_type": "normal",
      "storage_spec_code": "dms.physical.storage.normal"
    },
    {
      "io_type": "high",
      "storage_spec_code": "dms.physical.storage.high"
    },
    {
      "io_type": "ultra",
      "storage_spec_code": "dms.physical.storage.ultra"
    }
  ],
  "vm_specification": "16vCPUs 32GB",
  "product_id": "00300-30116-0--0",
```



```
    "spec_code": "dms.instance.rabbitmq.single.c3.16u32g"
  },
  "name": "single"
},
{
  "detail": [
    {
      "vm_specification": "4vCPUs 8GB",
      "product_info": [
        {
          "storage": "300",
          "io": [
            {
              "io_type": "normal",
              "storage_spec_code": "dms.physical.storage.normal"
            },
            {
              "io_type": "high",
              "storage_spec_code": "dms.physical.storage.high"
            },
            {
              "io_type": "ultra",
              "storage_spec_code": "dms.physical.storage.ultra"
            }
          ],
          "node_num": "3",
          "product_id": "00300-30210-0--0",
          "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.3"
        },
        {
          "storage": "500",
          "io": [
            {
              "io_type": "normal",
              "storage_spec_code": "dms.physical.storage.normal"
            },
            {
              "io_type": "high",
              "storage_spec_code": "dms.physical.storage.high"
            },
            {
              "io_type": "ultra",
              "storage_spec_code": "dms.physical.storage.ultra"
            }
          ],
          "node_num": "5",
          "product_id": "00300-30212-0--0",
          "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.5"
        },
        {
          "storage": "700",
          "io": [
            {
              "io_type": "normal",
              "storage_spec_code": "dms.physical.storage.normal"
            },
            {
              "io_type": "high",
              "storage_spec_code": "dms.physical.storage.high"
            },
            {
              "io_type": "ultra",
              "storage_spec_code": "dms.physical.storage.ultra"
            }
          ],
          "node_num": "7",
          "product_id": "00300-30214-0--0",
          "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.7"
        }
      ]
    }
  ]
}
```

```
    }
  ]
},
{
  "vm_specification": "8vCPUs 16GB",
  "product_info": [
    {
      "storage": "300",
      "io": [
        {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal"
        },
        {
          "io_type": "high",
          "storage_spec_code": "dms.physical.storage.high"
        },
        {
          "io_type": "ultra",
          "storage_spec_code": "dms.physical.storage.ultra"
        }
      ],
      "node_num": "3",
      "product_id": "00300-30216-0--0",
      "spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.3"
    },
    {
      "storage": "500",
      "io": [
        {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal"
        },
        {
          "io_type": "high",
          "storage_spec_code": "dms.physical.storage.high"
        },
        {
          "io_type": "ultra",
          "storage_spec_code": "dms.physical.storage.ultra"
        }
      ],
      "node_num": "5",
      "product_id": "00300-30218-0--0",
      "spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.5"
    },
    {
      "storage": "700",
      "io": [
        {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal"
        },
        {
          "io_type": "high",
          "storage_spec_code": "dms.physical.storage.high"
        },
        {
          "io_type": "ultra",
          "storage_spec_code": "dms.physical.storage.ultra"
        }
      ],
      "node_num": "7",
      "product_id": "00300-30220-0--0",
      "spec_code": "dms.instance.rabbitmq.cluster.c3.8u16g.7"
    }
  ]
},
{
  {
```

```
"vm_specification": "16vCPUs 32GB",
"product_info": [
  {
    "storage": "300",
    "io": [
      {
        "io_type": "normal",
        "storage_spec_code": "dms.physical.storage.normal"
      },
      {
        "io_type": "high",
        "storage_spec_code": "dms.physical.storage.high"
      },
      {
        "io_type": "ultra",
        "storage_spec_code": "dms.physical.storage.ultra"
      }
    ],
    "node_num": "3",
    "product_id": "00300-30222-0--0",
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.3"
  },
  {
    "storage": "500",
    "io": [
      {
        "io_type": "normal",
        "storage_spec_code": "dms.physical.storage.normal"
      },
      {
        "io_type": "high",
        "storage_spec_code": "dms.physical.storage.high"
      },
      {
        "io_type": "ultra",
        "storage_spec_code": "dms.physical.storage.ultra"
      }
    ],
    "node_num": "5",
    "product_id": "00300-30224-0--0",
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.5"
  },
  {
    "storage": "700",
    "io": [
      {
        "io_type": "normal",
        "storage_spec_code": "dms.physical.storage.normal"
      },
      {
        "io_type": "high",
        "storage_spec_code": "dms.physical.storage.high"
      },
      {
        "io_type": "ultra",
        "storage_spec_code": "dms.physical.storage.ultra"
      }
    ],
    "node_num": "7",
    "product_id": "00300-30226-0--0",
    "spec_code": "dms.instance.rabbitmq.cluster.c3.16u32g.7"
  }
]
},
"name": "cluster"
}
```

```
]
}
```

状态码

操作成功的状态码如[表7-33](#)所示，其他响应见[状态码](#)。

表 7-33 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

无。

响应消息

响应参数

参数说明见表[表7-34](#)、[表7-35](#)

表 7-34 响应参数说明

参数	类型	备注
maintain_windows	Array	支持的维护时间窗列表

表 7-35 maintain_windows 参数说明

参数	类型	备注
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": [
    {
      "seq": 1,
      "begin": "22",
      "end": "02",
      "default": false
    },
    {
      "seq": 2,
      "begin": "02",
      "end": "06",
      "default": true
    },
    {
      "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",
      "default": false
    }
  ]
}
```

状态码

操作成功的状态码如[表7-36](#)所示，其他响应见[状态码](#)。

表 7-36 状态码

状态码	描述
200	查询维护时间窗时间段成功。

7.1.2.4 查询配额

功能介绍

查询配额详情。

📖 说明

当前页面API为历史版本API，未来可能停止维护。

调用方法

请参见[如何调用API](#)。

URI

GET /v1.0/{project_id}/quotas/dms

表 7-37 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 。

请求参数

无

响应参数

状态码： 200

表 7-38 响应 Body 参数

参数	参数类型	描述
quotas	quotas object	配额信息。

表 7-39 quotas

参数	参数类型	描述
resources	Array of QuotaResourceEntity objects	配额列表。

表 7-40 QuotaResourceEntity

参数	参数类型	描述
type	String	支持rabbitmqInstance、kafkaInstance、rocketmqInstance、tags四种。 <ul style="list-style-type: none"> • rabbitmqInstance表示RabbitMQ实例配额。 • kafkaInstance表示Kafka实例配额。 • rocketmqInstance表示RocketMQ实例配额。 • tags表示标签的配额。
quota	Integer	租户最大可以创建的实例个数，或者每个实例最大可以创建的标签个数。
used	Integer	已创建的实例个数。

请求示例

GET https://{endpoint}/v1.0/{project_id}/quotas/dms

响应示例

状态码： 200

查询成功。

```
{
  "quotas": [
    {
      "resources": [
        {
          "type": "rabbitmqInstance",
          "quota": 100,
          "used": 3
        },
        {
          "type": "kafkaInstance",
          "quota": 100,
          "used": 17
        },
        {
          "type": "rocketmqInstance",
          "quota": 100,
          "used": 17
        },
        {
          "type": "tags",
          "quota": 20
        }
      ]
    }
  ]
}
```

```
}  
}
```

状态码

状态码	描述
200	查询成功。

错误码

请参见[错误码](#)。

7.1.2.5 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。

📖 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 /v1.0/dms/{project_id}/instances/{instance_id}/ceshierarchy

表 7-41 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

无

响应参数

状态码： 200

表 7-42 响应 Body 参数

参数	参数类型	描述
dimensions	Object	监控维度。
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 7-43 instance_ids

参数	参数类型	描述
name	String	实例ID。

表 7-44 nodes

参数	参数类型	描述
name	String	节点名称。

表 7-45 queues

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

表 7-46 partitions

参数	参数类型	描述
name	String	分区名称。

表 7-47 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 7-48 queues

参数	参数类型	描述
name	String	topic名称。
partitions	Array of partitions objects	分区信息

表 7-49 partitions

参数	参数类型	描述
name	String	分区名称。

请求示例

GET https://{endpoint}/v1.0/dms/{project_id}/instances/{instance_id}/ceshierarchy

响应示例

状态码： 200

查询成功。

```
{
  "dimensions": [ {
    "name": "rabbitmq_instance_id",
    "metrics": [ "connections", "channels", "queues", "consumers", "messages_ready",
"messages_unacknowledged", "publish", "deliver", "deliver_no_ack", "deliver_get", "instance_bytes_in_rate",
"instance_bytes_out_rate", "instance_disk_usage" ],
    "key_name": [ "instance_ids" ],
    "dim_router": [ "rabbitmq_instance_id" ]
  }, {
    "name": "rabbitmq_node",
    "metrics": [ "fd_used", "socket_used", "proc_used", "mem_used", "disk_free", "rabbitmq_alive",
"rabbitmq_disk_usage", "rabbitmq_cpu_usage", "rabbitmq_cpu_core_load", "rabbitmq_memory_usage",
"rabbitmq_disk_read_await", "rabbitmq_disk_write_await", "rabbitmq_node_bytes_in_rate",
"rabbitmq_node_bytes_out_rate", "rabbitmq_node_queues", "rabbitmq_memory_high_watermark",
"rabbitmq_disk_insufficient" ],
    "key_name": [ "nodes" ],
    "dim_router": [ "rabbitmq_instance_id", "rabbitmq_node" ]
  }, {
    "name": "rabbitmq_queue",
    "metrics": [ "queue_messages_unacknowledged", "queue_messages_ready" ],
    "key_name": [ "queues" ],
    "dim_router": [ "rabbitmq_instance_id", "rabbitmq_queue" ]
  } ]
}
```

```
  }],  
  "instance_ids" : [ {  
    "name" : "0e16280d-7451-4f5b-80fa-f210372ce657"  
  } ],  
  "nodes" : [ {  
    "name" : "dms-vm-0e16280d-rabbitmq-0",  
    "available_zone" : "cn-north-7c"  
  }, {  
    "name" : "dms-vm-0e16280d-rabbitmq-1",  
    "available_zone" : "cn-north-7c"  
  }, {  
    "name" : "dms-vm-0e16280d-rabbitmq-2",  
    "available_zone" : "cn-north-7c"  
  } ],  
  "queues" : [ {  
    "name" : "Vhost-17130843_Queue-21084756",  
    "vhost" : "default"  
  } ],  
  "vhosts" : [ {  
    "name" : "default"  
  } ],  
  "exchanges" : [ {  
    "name" : "direct_exchange",  
    "vhost" : "default"  
  } ],  
  "groups" : [ ]  
}
```

状态码

状态码	描述
200	查询成功。

错误码

请参见[错误码](#)。

7.2 API V2

7.2.1 查询产品规格列表

功能介绍

在创建实例时，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。

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](#)。

URI

GET /v2/products

表 7-50 Query 参数

参数	是否必选	参数类型	描述
engine	否	String	消息引擎的类型。当前只支持 rabbitmq。

请求参数

无

响应参数

状态码： 200

表 7-51 响应 Body 参数

参数	参数类型	描述
Hourly	Array of Hourly objects	表示按需付费的产品列表。
Monthly	Array of Monthly objects	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RabbitMQ实例。

表 7-52 Hourly

参数	参数类型	描述
name	String	消息引擎的名称，该字段显示为rabbitmq。
version	String	消息引擎的版本。
values	Array of values objects	产品规格列表。

表 7-53 values

参数	参数类型	描述
detail	Array of detail objects	规格详情。
name	String	实例类型。
unavailable_zones	Array of strings	资源售罄的可用区列表。

参数	参数类型	描述
available_zones	Array of strings	有可用资源的可用区列表。

表 7-54 detail

参数	参数类型	描述
storage	String	消息存储空间。
product_id	String	产品ID。
spec_code	String	规格ID。
io	Array of io objects	IO信息。
unavailable_zones	Array of strings	资源售罄的可用区列表。
available_zones	Array of strings	有可用资源的可用区列表。
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 7-55 io

参数	参数类型	描述
io_type	String	IO类型。
storage_spec_code	String	IO规格。
available_zones	Array of strings	IO未售罄的可用区列表。
unavailable_zones	Array of strings	IO已售罄的不可用区列表。
volume_type	String	磁盘类型。

表 7-56 Monthly

参数	参数类型	描述
name	String	消息引擎的名称，该字段显示为rabbitmq。

参数	参数类型	描述
version	String	消息引擎的版本。
values	Array of values objects	产品规格列表。

表 7-57 values

参数	参数类型	描述
detail	Array of detail objects	规格详情。
name	String	实例类型。
unavailable_zones	Array of strings	资源售罄的可用区列表。
available_zones	Array of strings	有可用资源的可用区列表。

表 7-58 detail

参数	参数类型	描述
storage	String	消息存储空间。
product_id	String	产品ID。
spec_code	String	规格ID。
io	Array of io objects	IO信息。
unavailable_zones	Array of strings	资源售罄的可用区列表。
available_zones	Array of strings	有可用资源的可用区列表。
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 7-59 io

参数	参数类型	描述
io_type	String	IO类型。

参数	参数类型	描述
storage_spec_code	String	IO规格。
available_zones	Array of strings	IO未售罄的可用区列表。
unavailable_zones	Array of strings	IO已售罄的不可用区列表。
volume_type	String	磁盘类型。

请求示例

GET https://{endpoint}/v2/products?engine=rabbitmq

响应示例

状态码： 200

查询规格列表成功。

```
{
  "Hourly": [ {
    "name": "RabbitMQ",
    "version": "3.x.x",
    "values": [ {
      "detail": [ {
        "storage": "100",
        "vm_specification": "2vCPUs 4GB",
        "product_id": "00300-30109-0--0",
        "spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
        "io": [ {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal",
          "available_zones": [ "cn-north-7b", "cn-north-7a", "cn-north-7c" ],
          "volume_type": "SATA"
        } ],
        "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
        "available_zones": [ "cn-north-7c" ],
        "ecs_flavor_id": "s6.medium.2",
        "arch_type": "X86"
      } ],
      "name": "single",
      "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
      "available_zones": [ "cn-north-7c" ]
    } ], {
      "detail": [ {
        "vm_specification": "4vCPUs 8GB",
        "product_info": [ {
          "storage": "300",
          "node_num": "3",
          "product_id": "00300-30209-0--0",
          "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.3",
          "io": [ {
            "io_type": "normal",
            "storage_spec_code": "dms.physical.storage.normal",
            "available_zones": [ "cn-north-7b", "cn-north-7a", "cn-north-7c" ],
            "volume_type": "SATA"
          } ],
          "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
          "available_zones": [ "cn-north-7c" ],
        } ],
      } ],
    } ],
  } ]
```

```

    "ecs_flavor_id": "c3.medium.4"
  }],
  "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
  "available_zones": [ "cn-north-7c" ],
  "arch_type": "X86"
}],
"name": "cluster",
"unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
"available_zones": [ "cn-north-7c" ]
}]
}],
"Monthly": [ {
  "name": "RabbitMQ",
  "version": "3.x.x",
  "values": [ {
    "detail": [ {
      "storage": "100",
      "vm_specification": "2vCPUs 4GB",
      "product_id": "00300-30110-0-0",
      "spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
      "io": [ {
        "io_type": "normal",
        "storage_spec_code": "dms.physical.storage.normal",
        "available_zones": [ "cn-north-7b", "cn-north-7a", "cn-north-7c" ],
        "volume_type": "SATA"
      } ],
      "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
      "available_zones": [ "cn-north-7c" ],
      "ecs_flavor_id": "s6.medium.2",
      "arch_type": "X86"
    } ],
    "name": "single",
    "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
    "available_zones": [ "cn-north-7c" ]
  } ], {
    "detail": [ {
      "vm_specification": "4vCPUs 8GB",
      "product_info": [ {
        "storage": "300",
        "node_num": "3",
        "product_id": "00300-30210-0-0",
        "spec_code": "dms.instance.rabbitmq.cluster.c3.4u8g.3",
        "io": [ {
          "io_type": "normal",
          "storage_spec_code": "dms.physical.storage.normal",
          "available_zones": [ "cn-north-7b", "cn-north-7a", "cn-north-7c" ],
          "volume_type": "SATA"
        } ],
        "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
        "available_zones": [ "cn-north-7c" ],
        "ecs_flavor_id": "c3.medium.4"
      } ],
      "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
      "available_zones": [ "cn-north-7c" ],
      "arch_type": "X86"
    } ],
    "name": "cluster",
    "unavailable_zones": [ "cn-north-7b", "cn-north-7a" ],
    "available_zones": [ "cn-north-7c" ]
  } ]
} ]
} ]
}

```


状态码

状态码	描述
200	查询规格列表成功。

错误码

请参见[错误码](#)。

7.2.2 RabbitMQ 实例内核升级（废弃）

功能介绍

RabbitMQ实例内核升级。

调用方法

请参见[如何调用API](#)。

URI

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

表 7-60 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。

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
job_id	String	后台任务ID

请求示例

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

响应示例

状态码： 200

查询成功。

```
{  
  "job_id" : "8a2c259182ab0e9d0182ab1882560011"  
}
```

状态码

状态码	描述
200	查询成功。

错误码

请参见[错误码](#)。

7.2.3 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 7-62 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 7-63 响应 Body 参数

参数	参数类型	描述
access_user	String	认证用户名，只能由英文字母、数字、中划线组成，长度为4~64的字符。
broker_num	Integer	代理个数。
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。 <ul style="list-style-type: none"> RabbitMQ实例单机返回vm规格。 RabbitMQ实例集群返回vm规格和节点数。
storage_space	Integer	消息存储空间，单位：GB。
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。 <ul style="list-style-type: none"> true：开启 false：未开启
connect_address	String	实例内网连接IP地址。
connect_domain_name	String	实例内网连接域名。
public_connect_address	String	实例公网连接IP地址。
public_connect_domain_name	String	实例公网连接域名。
port	Integer	实例连接端口。
status	String	实例的状态。
description	String	实例描述。
instance_id	String	实例ID。

参数	参数类型	描述
resource_spec_code	String	<p>资源规格标识。</p> <ul style="list-style-type: none"> dms.instance.rabbitmq.single.c3.2u4g: RabbitMQ单机, vm规格2u4g dms.instance.rabbitmq.single.c3.4u8g: RabbitMQ单机, vm规格4u8g dms.instance.rabbitmq.single.c3.8u16g: RabbitMQ单机, vm规格8u16g dms.instance.rabbitmq.single.c3.16u32g: RabbitMQ单机, vm规格16u32g dms.instance.rabbitmq.cluster.c3.4u8g.3: RabbitMQ集群, vm规格4u8g, 3个节点 dms.instance.rabbitmq.cluster.c3.4u8g.5: RabbitMQ集群, vm规格4u8g, 5个节点 dms.instance.rabbitmq.cluster.c3.4u8g.7: RabbitMQ集群, vm规格4u8g, 7个节点 dms.instance.rabbitmq.cluster.c3.8u16g.3: RabbitMQ集群, vm规格8u16g, 3个节点 dms.instance.rabbitmq.cluster.c3.8u16g.5: RabbitMQ集群, vm规格8u16g, 5个节点 dms.instance.rabbitmq.cluster.c3.8u16g.7: RabbitMQ集群, vm规格8u16g, 7个节点 dms.instance.rabbitmq.cluster.c3.16u32g.3: RabbitMQ集群, vm规格16u32g, 3个节点 dms.instance.rabbitmq.cluster.c3.16u32g.5: RabbitMQ集群, vm规格16u32g, 5个节点 dms.instance.rabbitmq.cluster.c3.16u32g.7: RabbitMQ集群, vm规格16u32g, 7个节点
charging_mode	Integer	<p>付费模式, 1表示按需计费, 0表示包年/包月计费。</p>
vpc_id	String	<p>VPC ID。</p>
vpc_name	String	<p>VPC的名称。</p>

参数	参数类型	描述
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。
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	RabbitMQ实例是否开启公网访问功能。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
publicip_address	String	RabbitMQ实例绑定的弹性IP地址。如果未开启公网访问功能，该字段值为 null。
publicip_id	String	RabbitMQ实例绑定的弹性IP地址的ID。如果未开启公网访问功能，该字段值为 null。
management_connect_address	String	RabbitMQ实例的管理地址。
management_connect_domain_name	String	RabbitMQ实例的管理域名。
public_management_connect_addresses	String	RabbitMQ实例的公网管理地址。
public_management_connect_domain_name	String	RabbitMQ实例的公网管理域名。
ssl_enable	Boolean	是否开启安全认证。 <ul style="list-style-type: none"> • true: 开启 • false: 未开启
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。 <ul style="list-style-type: none"> • 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。
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”。
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。
ipv6_enable	Boolean	是否开启ipv6。
ipv6_connect_addresses	Array of strings	IPv6的连接地址。
tags	Array of TagEntity objects	标签列表。

表 7-64 TagEntity

参数	参数类型	描述
key	String	标签键。 <ul style="list-style-type: none"> 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。 <ul style="list-style-type: none"> 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/{engine}/{project_id}/instances/{instance_id}

响应示例

状态码： 200

查询成功。

```
{
  "name": "api-explorer",
  "engine": "rabbitmq",
  "port": 5672,
  "status": "RUNNING",
  "type": "single",
  "specification": "2vCPUs 4GB",
  "engine_version": "3.8.35",
  "connect_address": "192.168.0.74",
  "instance_id": "de873040-d661-4770-aa96-9329c71d7c8a",
  "resource_spec_code": "dms.instance.rabbitmq.single.c3.2u4g",
  "charging_mode": 1,
  "vpc_id": "40a6501e-85ca-4449-a0db-b8bc7f0cec28",
  "vpc_name": "vpc-a400",
  "created_at": "1590047080687",
  "product_id": "00300-30109-0--0",
  "security_group_id": "bfd68e26-f8ef-4a91-a373-0a8f5c198601",
  "security_group_name": "Sys-default",
  "subnet_id": "a7f9a564-30dd-4059-8124-364ca6554578",
  "available_zones": [ "9f1c5806706d4c1fb0eb72f0a9b18c77" ],
  "user_id": "3df5acbc24a54fad62a043c9000a307",
  "user_name": "paas_dms_01",
  "maintain_begin": "22:00:00",
  "maintain_end": "02:00:00",
  "storage_space": 88,
  "total_storage_space": 100,
  "used_storage_space": 4,
}
```

```
"enable_publicip" : false,
"ssl_enable" : false,
"management_connect_address" : "http://192.168.0.74:15672",
"storage_resource_id" : "52be287d-1d6a-4d30-937e-185b3f176fc4",
"storage_spec_code" : "dms.physical.storage.normal",
"enterprise_project_id" : "0",
"tags" : [ {
  "key" : "key1",
  "value" : "value1"
}, {
  "key" : "key2",
  "value" : "value2"
} ],
"is_logical_volume" : true,
"extend_times" : 0,
"ipv6_enable" : false,
"ipv6_connect_addresses" : [ ],
"broker_num" : 1,
"access_user" : "root_0"
}
```

状态码

状态码	描述
200	查询成功。

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	Unprocessable Entity	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。

更多服务错误码请参见[API错误中心](#)。

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.0040001 2	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 letters Uppercase letters Digits Special 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.0040001 3	vpc_id in the request is empty.	请求参数 vpc_id 为空。	请检查参数 vpc_id
400	DMS.0040001 4	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.0040001 5	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.0040001 6	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 SmartConnect Task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.1024001 0	The description contains invalid characters.	描述包含无效字符。	请检查描述
400	DMS.1024001 1	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.1024001 2	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.1024001 3	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.1024001 4	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.1024001 5	The queue ID format is invalid.	Queue ID的格式无效。	请检查queueid
400	DMS.1024001 6	The group ID format is invalid.	Group ID的格式无效。	请检查groupid
400	DMS.1024001 7	The queue already exists.	队列已经存在。	请检查队列是否已存在
400	DMS.1024001 8	The consumer group already exists.	消费组已存在。	请检查消费组是否已存在
400	DMS.1024001 9	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.1024004 1	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.1024004 2	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.1024004 3	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.1024004 4	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.1024004 5	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.1054000 1	The message body contains invalid fields.	消息体的字段非法。	请检查消息体字段
400	DMS.1054000 3	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.1054000 4	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.1054001 0	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.1054001 1	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.1054001 2	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.1054001 4	The URL contains invalid parameters.	URI中参数错误。	请检查url参数
400	DMS.1054020 2	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.1054220 4	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.1054220 5	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.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.0040300 2	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.0040300 3	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.0040300 7	Authorization denied.	用户缺少权限，无法执行操作。	请检查用户权限
403	DMS.1024030 4	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.1024030 6	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.1024030 7	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.1024030 8	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.1024030 9	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.00500007	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.005000024	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.005000025	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.005000041	Internal service error.	内部服务错误。	请联系技术支持。
500	DMS.005000052	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
2025-02-14	本次变更如下： <ul style="list-style-type: none">在查询指定实例中，新增available_zone_names参数。
2024-12-20	本次变更如下： <ul style="list-style-type: none">新增查询RabbitMQ产品规格核数的API。
2024-07-17	本次变更如下： <ul style="list-style-type: none">新增Vhost管理、Exchange管理、Queue管理、Binding管理和用户管理的API。
2023-05-08	本次变更如下： <ul style="list-style-type: none">在创建实例中，新增bss_param参数，支持创建包周期实例。
2023-03-23	本次变更如下： <ul style="list-style-type: none">修改查询可扩容规格列表和实例规格变更接口，适配新规格实例的扩容。
2022-01-26	本次变更如下： <ul style="list-style-type: none">查询产品规格列表V2接口变更为新版本，旧版本接口移入历史API中。
2021-12-14	本次变更如下： <ul style="list-style-type: none">在权限和授权项中，将授权项明细接口从V1改为V2。
2020-08-25	新增以下接口： <ul style="list-style-type: none">查询可扩容规格列表实例规格变更

发布日期	修订记录
2020-08-18	新增以下接口： <ul style="list-style-type: none">● 重置密码● 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● 批量添加或删除实例标签● 查询实例标签● 查询项目标签
2020-08-06	本次变更如下： <ul style="list-style-type: none">● 将API V1移动至历史API。
2020-07-07	本次变更如下： <ul style="list-style-type: none">● 新增V2接口API。
2020-04-15	第一次正式发布。