对象存储服务

C SDK 开发指南

文档版本	01
发布日期	2025-02-18

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为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。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 使用前需知(C SDK)

本文介绍C SDK版本兼容性说明，以及其他使用前须知。

兼容性说明

- 3.*.* 相对于 3.0.0兼容
- 3.*.* 不兼容 2.*.*
- 3.*.* 不兼容 1.*.*
- arm编译环境：

```
NAME="EulerOS"
VERSION="2.0 (SP8)"
ID="euleros"
ID_LIKE="rhel fedora centos"
VERSION_ID="2.0"
PRETTY_NAME="EulerOS 2.0 (SP8)"
ANSI_COLOR="0;31"
```

内核版本：

```
4.19.36-vhulk1905.1.0.h276.eulerosv2r8.aarch64
```

gcc/g++版本：

```
gcc (GCC) 10.3.0/g++ (GCC) 10.3.0
```

- linux编译环境：

```
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"
```

内核版本：

```
3.10.0-957.5.1.el7.x86_64
```

gcc/g++版本：

```
gcc (GCC) 10.3.0/g++ (GCC) 10.3.0
```


说明

SDK二进制包编译环境如上，其余内核和操作系统的兼容性不做保证。如果有其它操作系统或者内核版本的需求，请使用开源代码自行编译。

使用前须知

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

- 请确认您已经熟悉OBS的基本概念，如**桶（Bucket）**、**对象（Object）**、**访问密钥（AK和SK）**等。
- 使用OBS客户端进行接口调用操作完成后，没有返回异常，则表明接口调用成功；如果返回异常，则说明操作失败，此时应从**SDK错误处理(C SDK)**中获取错误信息。
- 当前各区域特性开放不一致，部分特性只在部分区域开放，使用过程中如果接口HTTP状态码为405，请确认该区域是否支持该功能特性。

2 下载与安装 SDK(C SDK)

本节提供C SDK的下载链接，并介绍SDK的编译方式。

下载 SDK

- OBS C SDK最新版本源码：**最新源码下载**
- OBS C SDK历史版本下载：**历史版本下载**

SDK 编译

获取到**SDK源码**后，根据使用平台来开展编译工作。

- linux下：

linux可以直接进入到**source/eSDK_OBS_API/eSDK_OBS_API_C++/下执行下面的脚本来编译(x86和arm略有不同请注意)：**

export SPDLOG_VERSION=spdlog-1.12.0

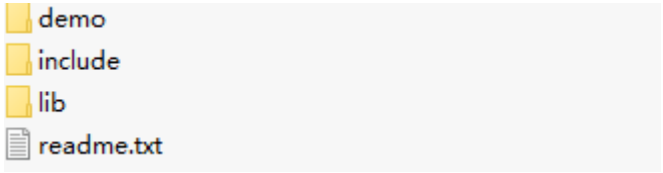
#x86执行这条命令

bash build.sh sdk

#arm执行这条命令

bash build_aarch.sh sdk

具体参数可见脚本内注释，生成产物为一个包含了demo代码，include，和lib的demo包（包名为sdk.tgz）。



- Windows下：

使用visual studio打开**source**/**eSDK_OBS_API**/**eSDK_OBS_API_C++**/**sln**/vc100/下的sln工程，生成obs项目，即可在输出目录（可在工程属性中查看）生成huaweisecurec.lib，huaweisecurec.dll,libeSDKOBS.lib和libeSDKOBS.dll。

3.24.3之前的版本的编译的详细步骤可以参考这个链接：**Windows下编译C SDK**

3 技术支持渠道(C SDK)

开发者社区提供的技术支持渠道如下：

- 开发过程中，您有任何问题可以在**华为云对象存储服务论坛**中发帖求助。

4 快速入门(C SDK)

4.1 OBS 服务环境搭建(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

步骤1 注册云服务账号

使用OBS之前必须要有一个云服务账号。

1. 打开浏览器。
2. 登录公有云网站**www.huaweicloud.com**。
3. 在页面右上角单击“注册”。
4. 按需填写注册信息并单击“同意协议并注册”。

步骤2 开通OBS服务

使用OBS服务之前必须先充值，才能正常使用OBS服务。

1. 登录OBS管理控制台。
2. 单击页面右上角的“费用”进入费用中心页面。
3. 单击“充值”，系统自动跳转到充值窗口。
4. 根据界面提示信息，对账户进行充值。
5. 充值成功后，关闭充值窗口，返回管理控制台首页。
6. 单击“对象存储服务”，开通并进入OBS管理控制台。

步骤3 创建访问密钥

OBS通过用户账号中的AK和SK进行签名验证，确保通过授权的账号才能访问指定的OBS资源。以下是对AK和SK的解释说明：

- AK：Access Key ID，接入键标识，用户在对象存储服务系统中的接入键标识，一个接入键标识唯一对应一个用户，一个用户可以同时拥有多个接入键标识。对象存储服务系统通过接入键标识识别访问系统的用户。

- SK：Secret Access Key，安全接入键，用户在对象存储服务系统中的安全接入键，是用户访问对象存储服务系统的密钥，用户根据安全接入键和请求头域生成鉴权信息。安全接入键和接入键标识一一对应。

访问密钥分永久访问密钥（AK/SK）和临时访问密钥（AK/SK和SecurityToken）两种。每个用户最多可创建两个有效的永久访问密钥。临时访问密钥只在设置的有效期内能够访问OBS，过期后需要重新获取。出于安全性考虑，建议您使用临时访问密钥访问OBS，或使用永久访问密钥访问OBS时，定期更新您的访问密钥（AK/SK）。两种密钥的获取方式如下。

- 永久访问密钥：
  a. 登录OBS控制台。
  b. 单击页面右上角的用户名，并选择“我的凭证”。
  c. 在“我的凭证”页面，单击左侧导航栏的“访问密钥”。
  d. 在“访问密钥”页面，单击“新增访问密钥”。
  e. 在弹出的“新增访问密钥”对话框中，输入登录密码和对应验证码。

     说明

     - 用户如果未绑定邮箱和手机，则只需输入登录密码。
     - 用户如果同时绑定了邮箱和手机，可以选择其中一种方式进行验证。

  f. 单击“确定”。
  g. 在弹出的“下载确认”提示框中，单击“确定”后，密钥会直接保存到浏览器默认的下载文件夹中。
  h. 打开下载下来的“credentials.csv”文件即可获取到访问密钥（AK和SK）。

     说明

     - 每个用户最多可创建两个有效的访问密钥。
     - 为防止访问密钥泄露，建议您将其保存到安全的位置。如果用户在此提示框中单击“取消”，则不会下载密钥，后续也将无法重新下载。如果需要使用访问密钥，可以重新创建新的访问密钥。

- 临时访问密钥：
  临时AK/SK和SecurityToken是系统颁发给用户的临时访问令牌，通过接口设置有效期，范围为15分钟至24小时，过期后需要重新获取。临时AK/SK和SecurityToken遵循权限最小化原则。使用临时AK/SK鉴权时，临时AK/SK和SecurityToken必须同时使用。
  获取临时访问密钥的接口请参考**获取临时AK/SK和securitytoken**。

须知

OBS属于全局级服务，所以在获取临时访问密钥时，需要设置Token的使用范围取值为domain，表示获取的Token可以作用于全局服务，全局服务不区分项目或者区域。

----结束

4.2 服务地址(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

- 您可以从**这里**查看OBS当前开通的服务地址和区域信息。

4.3 初始化 SDK(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

OBS客户端（ObsClient）是访问OBS服务的C客户端，它为调用者提供一系列与OBS服务进行交互的接口，用于管理、操作桶（Bucket）和对象（Object）等OBS服务上的资源。

使用OBS C SDK发起OBS请求，您需要先调用初始化接口，在进程退出的时候调用取消初始化的接口，释放资源。

在使用C SDK前要先调用初始化接口obs_initialize，而且进程中只需要调用一次：

```
obs_status  ret_status = OBS_STATUS_BUTT;
ret_status = obs_initialize(OBS_INIT_ALL);
if (OBS_STATUS_OK != ret_status)
{
    printf("obs_initialize failed(%s).\n", obs_get_status_name(ret_status));
    return ret_status;
}
obs_deinitialize();
// 请不要多次调用obs_initialize和obs_deinitialize，否则会导致程序访问无效的内存
```

4.4 初始化 option(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

在调用C SDK的功能函数时，都要传入option参数，您可通过init_obs_options函数初始化option配置，通过option设置AK、SK、Endpoint、bucket、超时时间、临时鉴权：

- 永久访问密钥（AK/SK）创建并初始化option如下：

```
// 创建并初始化option
obs_options option;
init_obs_options(&option);
option.bucket_options.host_name = "<your-endpoint>";
option.bucket_options.bucket_name = "<Your bucketname>";

// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
```

- 临时访问密钥（AK/SK和SecurityToken）创建并初始化option如下：

```
// 创建并初始化option
obs_options option;
init_obs_options(&option);
option.bucket_options.host_name = "<your-endpoint>";
option.bucket_options.bucket_name = "<Your bucketname>";

// 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
// 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
option.bucket_options.token = getenv("SecurityToken");
```

说明

- OBS属于全局级服务，所以在获取临时访问密钥时，需要设置Token的使用范围取值为domain，表示获取的Token可以作用于全局服务，全局服务不区分项目或者区域。

4.5 创建桶(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

桶是OBS全局命名空间，相当于数据的容器、文件系统的根目录，可以存储若干对象。以下代码展示如何新建一个桶：

```
static void test_create_bucket(obs_canned_acl canned_acl, char *bucket_name)
{
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建并初始化option
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucketname>";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    // 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");


    // 设置响应回调函数
    obs_response_handler response_handler =
```

```
    {
        0, &response_complete_callback
    };

    // 创建桶，<预定义访问策略>值参考5.5 管理桶访问权限
    create_bucket(&option, "<bucket ACL>", NULL, &response_handler, &ret_status);
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket successfully. \n");
    }
    else
    {
        printf("create bucket failed(%s).\n", obs_get_status_name(ret_status));
    }
}
```

说明

桶的名字是全局唯一的，所以您需要确保不与已有的桶名称重复。桶命名规则如下：

- 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。
- 禁止使用IP地址。
- 禁止以“-”或“.”开头及结尾。
- 禁止两个“.”相邻（如：“my..bucket”）。
- 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。
- 同一用户多次创建同名桶不会报错，创建的桶属性以第一次请求为准。

本示例创建的桶的访问权限默认是私有读写，存储类别默认是标准类型，区域位置为全局域名所在的默认区域。

更多创建桶的信息，请参见**创建桶(C SDK)**。

须知

- 创建桶时，如果使用的终端节点归属于默认区域华北-北京一（cn-north-1），则可以不指定区域；如果使用的终端节点归属于其他区域，则必须指定区域，且指定的区域必须与终端节点归属的区域一致。当前有效的区域名称可从**这里**查询。比如初始化时使用的终端节点EndPoint是obs.cn-north-4. myhuaweicloud.com，那么在创建桶的时候必须指定Location：cn-north-4 才会创建成功，否则会返回状态码400。

4.6 上传对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

数据流保存到callback_data（参见**流式上传(C SDK)**中的参数描述）中，使用结构体obs_put_object_handler中定义的回调函数put_object_data_callback把上传对象的内容复制到该回调函数的参数字符指针参数buffer中。

更多上传对象的信息，请参见**上传对象**。

以下代码展示了如何进行对象上传：

```
static void test_put_object_from_buffer()
{
    // 待上传buffer
    char *buffer = "abcdefg";
    // 待上传buffer的长度
    int buffer_size = strlen(buffer);
    // 上传的对象名
    char *key = "put_buffer_test";

    // 初始化option
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucket name>";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    // 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");

    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //自定义存储上传数据的结构体,
    put_buffer_object_callback_data data;
    memset(&data, 0, sizeof(put_buffer_object_callback_data));
    // 把buffer赋值到上传数据结构中
    data.put_buffer = buffer;
    // 设置buffersize
    data.buffer_size = buffer_size;
    // 设置回调函数,需要实现对应的回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_buffer_complete_callback },
          &put_buffer_data_callback
    };
    put_object(&option, key, buffer_size, &put_properties,0,&putobjectHandler,&data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from buffer successfully. \n");
    }
    else
    {
        printf("put object from buffer failed(%s).\n", obs_get_status_name(data.ret_status));
    }
}
```

4.7 下载对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

更多下载对象的信息，请参见**下载对象**。

以下代码展示了如何进行对象下载：

```
static void test_get_object()
{
    char *file_name = "./test";
    obs_object_info object_info;
    // 初始化option
```

```
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucketname>";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    // 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "<object key>";
    object_info.version_id = "<object version ID>";
    //根据业务需要,自定义存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 自定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback, &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&option, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
}
```

4.8 列举对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

以下代码展示了如何列举对象：

```
static void test_list_bucket_objects()
{
    // 创建并初始化option
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucketname>";

    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    // 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
```



```
    // 设置响应回调函数
    obs_list_objects_handler list_bucket_objects_handler =
    {
        { &response_properties_callback, &list_objects_complete_callback },
        &list_objects_callback
    };

    // 用户自定义回调数据
    list_bucket_callback_data data;
    memset(&data, 0, sizeof(list_bucket_callback_data));
    // 列举对象
    list_bucket_objects(&option, "<prefix>", "<marker>", "<delimiter>", "<maxkeys>",
&list_bucket_objects_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("list bucket objects successfully. \n");
    }
    else
    {
        printf("list bucket objects failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
}
```

说明

- 更多列举对象的信息，请参见：**列举桶内对象(C SDK)**

4.9 删除对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

以下代码展示了如何删除对象：

```
static void test_delete_object()
{
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建并初始化对象信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "<Your Key>";
    // 创建并初始化option
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucketname>";

    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    // 您可以登录访问管理控制台获取访问密钥AK/SK，获取方式请参见https://support.huaweicloud.com/usermanual-ca/ca_01_0003.html
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 设置响应回调函数
    obs_response_handler resqonseHandler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
```

```
    // 删除对象
    delete_object(&option,&object_info,&resqonseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("delete object successfully. \n");
    }
    else
    {
        printf("delete object failed(%s).\n", obs_get_status_name(ret_status));
    }
}
```

说明

- 更多删除对象的信息，请参见：**删除对象(C SDK)**

5 初始化(C SDK)

5.1 配置密钥(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

要接入OBS服务，您需要拥有一组有效的访问密钥（AK和SK）用来进行签名认证。具体可参考**OBS服务环境搭建(C SDK)**。

获取AK和SK之后，您便可以按照以下步骤进行初始化。

- **初始化SDK(C SDK)**
- **配置option(C SDK)**
- **配置SDK日志(C SDK)**

5.2 初始化 SDK(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

OBS客户端（ObsClient）是访问OBS服务的C客户端，它为调用者提供一系列与OBS服务进行交互的接口，用于管理、操作桶（Bucket）和对象（Object）等OBS服务上的资源。

使用OBS C SDK发起OBS请求，您需要先调用初始化接口，在进程退出的时候调用取消初始化的接口，释放资源。

在使用C SDK前要先调用初始化接口obs_initialize，而且进程中只需要调用一次：

```
obs_status  ret_status = OBS_STATUS_BUTT;
ret_status = obs_initialize(OBS_INIT_ALL);
if (OBS_STATUS_OK != ret_status)
{
    printf("obs_initialize failed(%s).\n", obs_get_status_name(ret_status));
    return ret_status;
}
obs_deinitialize();
// 请不要多次调用obs_initialize和obs_deinitialize，否则会导致程序访问无效的内存
```

5.3 配置 option(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

在调用C SDK的功能函数时，都要传入obs_options参数，您可通过init_obs_options函数初始化obs_options配置，通过obs_options设置AK、SK、Endpoint、bucket、超时时间、临时鉴权。obs_options主要包括obs_bucket_context和obs_http_request_option两个结构，可以设置的参数见下表：

表 5-1 obs_options.obs_bucket_context 参数

参数	描述	默认值	建议值
host_name	请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。	NULL	-
bucket_name	操作的桶名。	NULL	-
protocol	请求使用的协议类型: http、https。（出于安全性考虑，建议使用https协议）	HTTPS协议：OBS_PROTOCOL_HTTPS	OBS_PROTOCOL_HTTPS
access_key	连接对象存储服务的AK	NULL	-
secret_access_key	鉴权使用的SK，可用于字符串的签名。	NULL	-
obs_storage_class	在PUT，POST请求中，需要配置存储类别时设置此参数。	标准存储：OBS_STORAGE_CLASS_STANDARD	默认值
token	临时访问密钥的SecurityToken。	NULL	-
bucket_type	创桶时，指定是对象桶还是并行文件系统	对象桶：OBS_BUCKET_OBJECT	-

参数	描述	默认值	建议值
bucket_list_type	列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统	所有桶：OBS_BUCKET_LIST_ALL	-

表 5-2 obs_options.obs_http_request_option 参数

参数	描述	默认值	建议值
connect_time	建立HTTP/HTTPS连接的超时时间（单位：毫秒）。默认为60000毫秒。	60000	[10000, 60000]
max_connected_time	请求超时时间（单位：秒）。0代表永远不会断开链接。	0	0
proxy_auth	代理认证信息，格式username:password	NULL	-
proxy_host	代理服务器	NULL	-

说明

如网络状况不佳，建议增大connect_time和max_connected_time的值。

5.4 配置 SDK 日志(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

OBS C SDK的日志路径是通过OBS.ini中LogPath字段指定的，日志默认生成于与C SDK动态库lib目录同级的logs目录中，OBS.ini文件应与动态库（libeSDKLogAPI.so）同一目录。

OBS C SDK支持通过set_obs_log_path来指定日志路径，该方法存在两个参数，第一个参数为指定的路径，第二个参数为判断设定路径方式的信号，当该信号为True时，SDK将在第一个参数给定的路径下寻找OBS.ini进行日志配置，当该信号为Fasle时，SDK将在第一个参数给定的路径下生成OBS.ini和日志文件。

说明

- 您可以从**日志分析(C SDK)**章节获取更多关于SDK日志的信息。

6 管理桶(C SDK)

6.1 创建桶(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS桶是对象的容器，您上传的文件都将以对象的形式存放在桶中。本文介绍如何使用C SDK创建桶。

调用创建桶接口，将在当前账号按照用户指定的桶名创建一个新桶，接口支持根据用户诉求，在创建桶的同时配置桶的存储类别、区域及桶的访问权限等参数。

接口约束

- 您必须拥有obs:bucket:CreateBucket权限，才能创建桶。建议使用IAM进行授权，配置方式详见**使用IAM自定义策略**。
- OBS支持的region以及region与endPoint的对应关系，详细信息请参见**地区与终端节点**。

  创建桶时，如果初始化客户端使用的终端节点（endPoint）为“obs.myhuaweicloud.com”，则可以不指定桶所在区域（location），系统会自动在华北-北京一（cn-north-1）创建桶；如果初始化客户端使用的终端节点（endPoint）不是obs.myhuaweicloud.com，则必须指定桶所在区域（location），且指定的区域必须与终端节点（endPoint）区域一致，否则会返回状态码400。

  比如初始化时使用的终端节点endPoint是obs.cn-north-4. myhuaweicloud.com，那么在创建桶的时候必须指定Location：cn-north-4 才会创建成功，否则会返回状态码400。
- 同一账号下，可以创建多个存桶，数量上限是100个（不区分地域），存储桶中的对象数量和大小没有限制。

- 新创建桶的桶名在OBS中必须是唯一的。如果是同一个用户重复创建同一区域的同名桶时返回HTTP状态码200。除此以外的其他场景重复创建同名桶返回HTTP状态码409，表明桶已存在。
- 用户删除桶后，需要等待30分钟才能创建同名桶和并行文件系统。
- 并不是所有区域都支持创建多AZ桶，您可以在**产品价格详情**页面，查询指定区域是否支持多AZ。

方法定义

```
void create_bucket(const obs_options *options, obs_canned_acl canned_acl,
                   const char *location_constraint, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 6-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
canned_acl	**obs_canned_acl**	必选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。

参数名称	参数类型	是否必选	描述
location_constraint	const char *	必选	**参数解释：** 桶所在的区域。 **约束限制：** 该参数定义了桶将会被创建在哪个区域，如果使用的终端节点是obs.myhuaweicloud.com，可以不携带此参数；如果使用的终端节点不是obs.myhuaweicloud.com，则必须携带此参数。 **取值范围：** 当前有效的OBS区域和终端节点的更多信息，请参考**地区和终端节点**。终端节点即调用API的请求地址，不同服务不同区域的终端节点不同，您可以向企业管理员获取区域和终端节点信息。 **默认取值：** 终端节点为obs.myhuaweicloud.com且用户未设定区域时，默认为华北-北京一（cn-north-1）。
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-3 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。

枚举值	说明
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 6-4 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把obs_response_properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-5 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 6-6 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-7 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-8 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-9 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-10 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-11 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-12 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-13 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-14 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据)中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 6-15 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 该头域表示解密使用的算法。 **约束限制：** 如果服务端加密是SSE-C方式，响应包含该头域。 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 该头域表示解密使用的密钥的MD5值。 **约束限制：** • 如果服务端加密是SSE-C方式，响应包含该头域。 • 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-16 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-17 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-18 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：新建对象桶

以下示例展示如何新建一个对象桶：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何新建一个对象桶：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);

    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    // 设置桶区域位置(以区域为华北-北京四为例)，bucketLocation需要与endpoint的位置信息一致
    char * bucketLocation = "cn-north-4";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    // 设置桶的存储类别，此处以标准存储为例
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_STANDARD;
```



```
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建桶，并设置桶的ACL权限，此处以设置桶为私有为例
    create_bucket(&options, OBS_CANNED_ACL_PRIVATE, bucketLocation,
                  &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket %s successfully. \n", bucketName);
    } else {
        printf("create bucket %s failed(%s).\n", bucketName, obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
```



```
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例：新建 3AZ 的对象桶

以下示例展示如何新建一个3AZ的对象桶：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何新建一个3AZ的对象桶：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
```



```
    // 设置桶区域位置(以区域为华北-北京四为例)，bucketLocation需要与endpoint的位置信息一致
    char * bucketLocation = "cn-north-4";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    // 设置桶的存储类别，此处以标准存储为例
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_STANDARD;
    obs_create_bucket_params create_param;
    // 设置桶访问权限
    create_param.canned_acl = OBS_CANNED_ACL_PRIVATE;
    // 设置桶区域位置
    create_param.location_constraint = bucketLocation;
    // 设置桶的数据冗余策略，此处以3AZ为例
    create_param.az_redundancy = OBS_REDUNDANCY_3AZ;
    obs_status ret_status = OBS_STATUS_BUTT;
    // 新建一个3AZ的对象桶
    create_bucket_with_params(&options, &create_param, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket %s successfully. \n", bucketName);
    } else {
        printf("create bucket %s failed(%s).\n", bucketName, obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
```



```
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例：新建并行文件系统

以下示例展示如何新建一个并行文件系统的桶：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何新建一个并行文件系统的桶：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
```



```
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "example-posix-bucket-name";
    // 设置桶区域位置(以区域为华北-北京四为例)，bucketLocation需要与endpoint的位置信息一致
    char * bucketLocation = "cn-north-4";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建并行文件系统的桶
    create_pfs_bucket(&options, OBS_CANNED_ACL_PRIVATE, bucketLocation, &response_handler,
&ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket successfully. \n");
    }
    else
    {
        printf("create bucket failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
```



```
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例：带参数创建

以下示例展示创建桶时指定桶的访问权限、存储类别和区域位置：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
```



```
int main()
{
    // 以下示例展示创建桶时指定桶的访问权限、存储类别和区域位置:
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    // 设置桶区域位置(以区域为华北-北京四为例)，bucketLocation需要与endpoint的位置信息一致
    char * bucketLocation = "cn-north-4";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    // 设置桶的存储类别，此处以标准存储为例
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_STANDARD;
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建桶时指定桶的访问权限为私有(OBS_CANNED_ACL_PRIVATE)、存储类别
(OBS_STORAGE_CLASS_STANDARD)和区域位置(bucketLocation):
    create_bucket(&options, OBS_CANNED_ACL_PRIVATE, bucketLocation,
                  &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket %s successfully. \n", bucketName);
    } else {
        printf("create bucket %s failed(%s).\n", bucketName, obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
```



```
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于创建桶的API说明，请参见**创建桶**。

- 更多关于创建桶的代码示例，请参见**Github示例**。
- 创建桶过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 创建桶常见问题请参见**创建桶失败**。

6.2 列举桶(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS桶是对象的容器，您上传的文件都存放在桶中。调用列举桶接口，可列举当前账号所有地域下符合指定条件的桶。获取到的桶列表将按照桶名的字典顺序排列。

接口约束

- 您必须拥有obs:bucket:ListAllMyBuckets权限，才能获取桶列表。建议使用IAM进行授权，配置方式详见**使用IAM自定义策略**。
- OBS支持的region以及region与endPoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void list_bucket_obs(const obs_options *options, obs_list_service_obs_handler *handler, void *callback_data);
```

请求参数说明

表 6-19 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过**obs_options**设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_list_service_obs_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-20 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-21 obs_list_service_obs_handler

参数名称	参数类型	是否必选	描述
response_handler	obs_response_handler *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
listServiceCallback	obs_list_service_obs_callback *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-22 obs_list_service_obs_callback

参数名称	参数类型	是否必选	描述
owner_id	const char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	const char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
creation_date_seconds	int64_t	必选	**参数解释：** 桶的创建时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
location	const char *	必选	**参数解释：** 桶所在的区域。 **约束限制：** 该参数定义了桶将会被创建在哪个区域，如果使用的终端节点是obs.myhuaweicloud.com，可以不携带此参数；如果使用的终端节点不是obs.myhuaweicloud.com，则必须携带此参数。 **取值范围：** 当前有效的OBS区域和终端节点的更多信息，请参考**地区和终端节点**。终端节点即调用API的请求地址，不同服务不同区域的终端节点不同，您可以向企业管理员获取区域和终端节点信息。 **默认取值：** 终端节点为obs.myhuaweicloud.com且用户未设定区域时，默认为华北-北京一（cn-north-1）。
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-23 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-24 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-25 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-26 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-27 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-28 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-29 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-30 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-31 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中 **约束限制：** 无

表 6-32 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-33 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-34 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** 请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1' **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 该头域表示解密使用的算法。 **约束限制：** 如果服务端加密是SSE-C方式，响应包含该头域。 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-35 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-36 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-37 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：列举桶列表

本示例用于列举桶列表：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void list_bucket_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
// 自定义列举桶回调结构体
typedef struct list_service_data
{
    int headerPrinted;
    int allDetails;
    obs_status ret_status;
} list_service_data;
obs_status listServiceObsCallback(const char *owner_id, const char *bucket_name,
    int64_t creationDate, const char *location, void *callback_data);
int main()
{
    // 以下示例展示如何列举对象桶：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");

    list_service_data data;
```



```
    memset(&data, 0, sizeof(data));
    // 设置为1时增加打印桶的Owner ID
    data.allDetails = 1;
    // 自定义响应回调函数
    obs_list_service_obs_handler listHandler =
    {
        {response_properties_callback,
        &list_bucket_complete_callback },
        &listServiceObsCallback
    };
    // 列举桶
    list_bucket_obs(&options, &listHandler, &data);
    if (data.ret_status == OBS_STATUS_OK)
    {
        printf("list bucket successfully. \n");
    }
    else
    {
        printf("list bucket failed(%s).\n", obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
```



```
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void list_bucket_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        list_service_data *data = (list_service_data *)callback_data;
        data->ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void printListServiceObsHeader(int allDetails)
{
    printf("%-56s  %-20s  %-20s", "                        Bucket",
        "      Created",
        "     Location");
    if (allDetails) {
        printf("  %-64s",
            "                           Owner ID");
    }
    printf("\n");
    printf("--------------------------------------------------------  "
        "--------------------  "
        "--------------------");
    if (allDetails) {
        printf("  ----------------------------------------------------------------");
    }
    printf("\n");
}
obs_status listServiceObsCallback(const char *owner_id,
    const char *bucket_name,
    int64_t creationDate,
    const char *location,
    void *callback_data)
```

```
{
    list_service_data *data = (list_service_data *)callback_data;
    if (!data->headerPrinted) {
        data->headerPrinted = 1;
        printListServiceObsHeader(data->allDetails);
    }
    char timebuf[256] = { 0 };
    if (creationDate >= 0) {
        time_t t = (time_t)creationDate;
        strftime(timebuf, sizeof(timebuf), "%Y-%m-%dT%H:%M:%SZ", gmtime(&t));
    }
    else {
        timebuf[0] = 0;
    }
    printf("%-56s  %-20s  %-20s", bucket_name, timebuf, location);
    if (data->allDetails) {
        printf("  %-64s", owner_id ? owner_id : "");
    }
    printf("\n");
    return OBS_STATUS_OK;
}
```

相关链接

- 关于列举桶的API说明，请参见**创建桶**。
- 更多关于列举桶的代码示例，请参见**Github示例**。
- 列举桶过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

6.3 删除桶(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

桶为空时，用户可以删除桶，以免占用桶数量配额。删除桶后需要等待30分钟才能创建同名桶。

说明

华为云无法恢复用户主动删除的OBS数据。因此调用接口删除桶后，桶相关数据无法恢复，请谨慎操作。

接口约束

- 待删除的桶必须为空，桶为空包含两方面含义：
  - 桶内没有任何对象，没有对象的任何历史版本，没有对象的删除标记（删除标记也视作一个历史版本）。
  - 桶内没有任何未合并的多段上传任务，即桶内不存在碎片。
- 您必须是桶拥有者或拥有删除桶的权限，才能删除桶。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:DeleteBucket权限，如果使用桶策略则需授予DeleteBucket权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。

- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void delete_bucket(const obs_options *options, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 6-38 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-39 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-40 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback*	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-41 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-42 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-43 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-44 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-45 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-46 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-47 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-48 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-49 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-50 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-51 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-52 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-53 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-54 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：删除桶

本示例用于删除example-bucket-name桶。

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何删除桶：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
```



```
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 删除桶
    delete_bucket(&options, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("delete bucket successfully. \n");
    }
    else
    {
        printf("delete bucket failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
```



```
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

说明

- 如果桶不为空（包含对象或分段上传碎片），则该桶无法删除。
- 删除桶非幂等操作，删除不存在的桶会报错。

相关链接

- 如何删除桶内对象和历史版本，请参见**删除OBS桶中的对象**。
- 如何清理碎片，请参见**清理碎片**。
- 您可以使用**列举桶内对象**和**列举多段上传任务**来确认桶是否为空。
- 关于删除桶的API说明，请参见**删除桶**。
- 更多关于删除桶的代码示例，请参见**Github示例**。
- 删除桶过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 删除桶常见问题请参见**删除桶失败**。

6.4 判断桶是否存在(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

判断指定桶名的桶是否存在，返回的结果中HTTP状态码为200表明桶存在，否则返回404表明桶不存在。

接口约束

- 您必须是桶拥有者或拥有判断桶是否存在的权限，才能判断桶是否存在。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:HeadBucket权限，如果使用桶策略则需授予HeadBucket权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void obs_head_bucket(const obs_options *options, obs_response_handler *handler,
                     void *callback_data);
```

请求参数说明

表 6-55 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-56 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-57 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-58 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 6-59 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-60 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-61 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-62 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-63 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-64 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-65 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-66 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-67 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-68 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-69 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-70 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-71 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：判断桶是否存在

本示例用于判断example-bucket-name桶是否存在。

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何判断该桶是否已存在：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
```



```
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 判断桶是否存在
    obs_head_bucket(&options, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK)
    {
        printf("head bucket successfully. \n");
    }
    else
    {
        printf("head bucket failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
```



```
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 判断桶是否存在和获取桶元数据是同一个REST API，关于判断桶是否存在的API说明，请参见**获取桶元数据**。
- 判断桶是否存在过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

6.5 设置桶 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持对桶操作进行权限控制，您可以为桶设置访问策略，指定某一个用户对某一个桶是否有权行使某一项指定操作。OBS权限控制的方式有IAM、桶策略和ACL三种，ACL按照粒度又分为桶ACL和对象ACL，本节将对桶ACL接口进行详细介绍，更多权限相关内容可参见《对象存储服务权限配置指南》的**OBS权限控制概述**章节。

桶ACL是跨账号场景的权限，设置授权的对象不是当前账号，也不是当前账号下的IAM用户，而是另一个华为云账号及其账号下的IAM用户；授权的范围是以桶为粒度的，一条ACL策略为一个桶设置策略，因此设置ACL策略时您必须明确指定桶名；桶ACL授予的权限包括桶的访问权限和桶ACL的访问权限两个方面，桶的访问权限包括对桶及桶内对象的查看和编辑权限，桶ACL的访问权限包括对桶ACL策略的查看和编辑权限，详情可参见**ACL权限控制方式介绍**。

调用设置桶ACL接口，您可以修改指定桶的ACL策略。

接口约束

- 单个桶最多支持100条ACL策略。
- 您必须是桶拥有者或拥有设置桶ACL的权限，才能设置桶ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketAcl权限，如果使用桶策略则需授予PutBucketAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_acl(const obs_options * options, manager_acl_info * aclinfo,
                    obs_response_handler * handler, void *callback_data);
```

请求参数说明

表 6-72 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
aclinfo	**manager_acl_info** *	必选	**参数解释：** 访问策略结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-73 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-74 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号，如果非多版本对象，请把version设置为NULL。 **约束限制：** 无
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。

参数名称	参数类型	是否必选	描述
acl_grants	obs_acl_grant *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 6-75 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-76 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 6-77 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	**obs_grantee_type**	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见**obs_grantee_type**。
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递，有效值。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指桶的ACL不向桶内对象传递）

表 6-78 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 6-79 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 6-80 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

表 6-81 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-82 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-83 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-84 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-85 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-86 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-87 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-88 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-89 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-90 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-91 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 6-92 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-93 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-94 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-95 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例一：创桶时指定预定义访问策略

以下示例展示如何在创建桶时指定预定义访问策略：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何在创建桶时指定预定义访问策略：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    // 设置桶区域位置(以区域为华北-北京四为例)，bucketLocation需要与endpoint的位置信息一致
    char * bucketLocation = "cn-north-4";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
```



```
    &response_properties_callback,
    &response_complete_callback
};
obs_status  ret_status = OBS_STATUS_BUTT;
// 创建桶，并设置桶的ACL权限，此处以设置桶为私有为例
create_bucket(&options, OBS_CANNED_ACL_PRIVATE, bucketLocation, &response_handler, &ret_status);
// 判断请求是否成功
if (ret_status == OBS_STATUS_OK) {
    printf("create bucket successfully. \n");
}
else
{
    printf("create bucket failed(%s).\n", obs_get_status_name(ret_status));
}
// 释放分配的全局资源
obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
if (properties == NULL)
{
    printf("error! obs_response_properties is null!");
    if (callback_data != NULL)
    {
        obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
        printf("server_callback buf is %s, len is %llu",
            data->buffer, data->buffer_len);
        return OBS_STATUS_OK;
    }
    else {
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                              \
do {                                                    \
    if (properties-> field) {                               \
        printf("%s: %s\n", name, properties->field);        \
    }                                                   \
} while (0)
print_nonnull("request_id", request_id);
print_nonnull("request_id2", request_id2);
print_nonnull("content_type", content_type);
if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
}
print_nonnull("server", server);
print_nonnull("ETag", etag);
print_nonnull("expiration", expiration);
print_nonnull("website_redirect_location", website_redirect_location);
print_nonnull("version_id", version_id);
print_nonnull("allow_origin", allow_origin);
print_nonnull("allow_headers", allow_headers);
print_nonnull("max_age", max_age);
print_nonnull("allow_methods", allow_methods);
print_nonnull("expose_headers", expose_headers);
print_nonnull("storage_class", storage_class);
print_nonnull("server_side_encryption", server_side_encryption);
print_nonnull("kms_key_id", kms_key_id);
print_nonnull("customer_algorithm", customer_algorithm);
print_nonnull("customer_key_md5", customer_key_md5);
print_nonnull("bucket_location", bucket_location);
print_nonnull("obs_version", obs_version);
print_nonnull("restore", restore);
print_nonnull("obs_object_type", obs_object_type);
print_nonnull("obs_next_append_position", obs_next_append_position);
print_nonnull("obs_head_epid", obs_head_epid);
```



```
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例二：为已创建的桶设置预定义访问策略

以下示例展示如何为已创建的桶设置预定义访问策略：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何通过相关头域为桶设置ACL访问策略
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
```



```
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 设置桶预定义访问策略
    obs_canned_acl canned_acl = OBS_CANNED_ACL_PUBLIC_READ_WRITE;
    set_bucket_acl_by_head(&options, canned_acl, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set bucket acl by head successfully. \n");
    }
    else
    {
        printf("set bucket acl by head failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
```



```
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例三：直接设置桶访问权限

以下示例展示如何直接设置桶访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
void init_acl_info(manager_acl_info *aclinfo);
void deinitialize_acl_info(manager_acl_info *aclinfo);
int main()
{
    // 以下示例展示如何通过body体为桶设置ACL访问策略
```

```
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 创建并初始化acl信息
    manager_acl_info aclinfo;
    init_acl_info(&aclinfo);
    // 设置桶拥有者ID (owner_id)
    strcpy(aclinfo.owner_id, "1*****************************a");
    // 设置被授权账号的租户ID(canonical_user.id)
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.id, "0*****************************0");
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.display_name, "name1");
    aclinfo.acl_grants[0].grantee_type = OBS_GRANTEE_TYPE_CANONICAL_USER;
    // 设置ACL权限，此处以读权限为例
    aclinfo.acl_grants[0].permission = OBS_PERMISSION_READ;
    // 设置桶的ACL是否向桶内对象传递，此处设置为不传递
    aclinfo.acl_grants[0].bucket_delivered = BUCKET_DELIVERED_FALSE;
    // 设置桶的访问权限
    set_bucket_acl(&options, &aclinfo, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket acl successfully. \n");
    }
    else
    {
        printf("set bucket acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放内存
    deinitialize_acl_info(&aclinfo);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
```



```
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
```



```
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void init_acl_info(manager_acl_info *aclinfo)
{
    memset(aclinfo, 0, sizeof(manager_acl_info));
    aclinfo->acl_grants = (obs_acl_grant*)malloc(sizeof(obs_acl_grant));
    memset(aclinfo->acl_grants, 0, sizeof(obs_acl_grant));
    strcpy(aclinfo->acl_grants->grantee.canonical_user.id, "userid1");
    strcpy(aclinfo->acl_grants->grantee.canonical_user.display_name, "name1");
    aclinfo->acl_grants->grantee_type = OBS_GRANTEE_TYPE_LOG_DELIVERY;
    aclinfo->acl_grants->permission = OBS_PERMISSION_WRITE;
    aclinfo->acl_grants->bucket_delivered = BUCKET_DELIVERED_FALSE;
    aclinfo->acl_grant_count_return = (int*)malloc(sizeof(int));
    *(aclinfo->acl_grant_count_return) = 1;
    aclinfo->owner_id = (char *)malloc(sizeof(char) * OBS_MAX_GRANTEE_USER_ID_SIZE);
    memset(aclinfo->owner_id, 0, sizeof(char) * OBS_MAX_GRANTEE_USER_ID_SIZE);
    aclinfo->owner_display_name = (char *)malloc(sizeof(char) * OBS_MAX_GRANTEE_USER_ID_SIZE);
    memset(aclinfo->owner_display_name, 0, sizeof(char) * OBS_MAX_GRANTEE_USER_ID_SIZE);
    memset(&aclinfo->object_info, 0, sizeof(aclinfo->object_info));
}
void deinitialize_acl_info(manager_acl_info *aclinfo)
{
    free(aclinfo->acl_grants);
    free(aclinfo->owner_display_name);
    free(aclinfo->owner_id);
    free(aclinfo->acl_grant_count_return);
}
```

相关链接

- 关于设置桶ACL的API说明，请参见**设置桶ACL**。
- 更多关于设置桶ACL的代码示例，请参见**Github示例**。
- 设置桶ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 权限相关常见问题请参见**权限相关常见问题**。

6.6 获取桶 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持对桶操作进行权限控制，您可以为桶设置访问策略，指定某一个用户对某一个桶是否有权行使某一项指定操作。OBS权限控制的方式有IAM、桶策略和ACL三种，ACL按照粒度又分为桶ACL和对象ACL，本节将对桶ACL接口进行详细介绍，更多权限相关内容可参见《对象存储服务权限配置指南》的**OBS权限控制概述**章节。

桶ACL是跨账号场景的权限，设置授权的对象不是当前账号，也不是当前账号下的IAM用户，而是另一个华为云账号及其账号下的IAM用户；授权的范围是以桶为粒度的，

一条ACL策略为一个桶设置策略，因此设置ACL策略时您必须明确指定桶名；桶ACL授予的权限包括桶的访问权限和桶ACL的访问权限两个方面，桶的访问权限包括对桶及桶内对象的查看和编辑权限，桶ACL的访问权限包括对桶ACL策略的查看和编辑权限，详情可参见**ACL权限控制方式介绍**。

调用获取桶ACL接口，您可以获取指定桶的ACL策略。

接口约束

- 您必须是桶拥有者或拥有获取桶ACL的权限，才能获取桶ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketAcl权限，如果使用桶策略则需授予GetBucketAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_acl(const obs_options * options, manager_acl_info * aclinfo,
        obs_response_handler * handler, void *callback_data);
```

请求参数说明

表 6-96 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
aclinfo	**manager_acl_info** *	必选	**参数解释：** 访问策略结构体。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-97 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-98 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号，如果非多版本对象，请把version设置为NULL。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。
acl_grants	**obs_acl_grant** *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 6-99 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-100 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 6-101 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	obs_grantee_type	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见obs_grantee_type。

参数名称	参数类型	是否必选	描述
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递，有效值。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指桶的ACL不向桶内对象传递）

表 6-102 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 6-103 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 6-104 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

表 6-105 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-106 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-107 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-108 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-109 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-110 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-111 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-112 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-113 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-114 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-115 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 6-116 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-117 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-118 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-119 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶访问权限

您可以通过get_bucket_acl获取桶的访问权限。以下示例展示如何获取桶访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
manager_acl_info* malloc_acl_info();
void free_acl_info(manager_acl_info **acl);
void print_grant_info(int acl_grant_count, obs_acl_grant *acl_grants);
int main()
{
    // 以下示例展示如何获取桶访问权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
```



```
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 申请acl结构内存
    manager_acl_info *aclinfo = malloc_acl_info();
    // 调用获取权限接口
    get_bucket_acl(&options, aclinfo, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("get bucket acl: ----------");
        printf("%s\n", aclinfo->owner_id);
        if (aclinfo->acl_grant_count_return)
        {
            print_grant_info(*aclinfo->acl_grant_count_return, aclinfo->acl_grants);
        }
    }
    else
    {
        printf("get bucket acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放内存
    free_acl_info(&aclinfo);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
```



```
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
manager_acl_info* malloc_acl_info()
{
    manager_acl_info *aclinfo = (manager_acl_info*)malloc(sizeof(manager_acl_info));
    memset(aclinfo, 0, sizeof(manager_acl_info));
    aclinfo->acl_grants = (obs_acl_grant*)malloc(sizeof(obs_acl_grant) * OBS_MAX_ACL_GRANT_COUNT);
    memset(aclinfo->acl_grants, 0, sizeof(obs_acl_grant) * OBS_MAX_ACL_GRANT_COUNT);
    aclinfo->acl_grant_count_return = (int*)malloc(sizeof(int));
    *(aclinfo->acl_grant_count_return) = OBS_MAX_ACL_GRANT_COUNT;
    size_t owner_id_size = (OBS_MAX_GRANTEE_USER_ID_SIZE + 1) * sizeof(char);
    aclinfo->owner_id = (char *)malloc(owner_id_size);
    memset(aclinfo->owner_id, 0, owner_id_size);
    aclinfo->owner_display_name = (char *)malloc(OBS_MAX_GRANTEE_DISPLAY_NAME_SIZE);
    memset(aclinfo->owner_display_name, 0, OBS_MAX_GRANTEE_DISPLAY_NAME_SIZE);
    return aclinfo;
}
void free_acl_info(manager_acl_info **acl)
{
```



```
    manager_acl_info *aclinfo = *acl;
    free(aclinfo->acl_grants);
    free(aclinfo->owner_display_name);
    free(aclinfo->owner_id);
    free(aclinfo->acl_grant_count_return);
    free(aclinfo);
}
void print_grant_info(int acl_grant_count, obs_acl_grant *acl_grants)
{
    int i;
    for (i = 0; i < acl_grant_count; i++)
    {
        obs_acl_grant *grant = acl_grants + i;
        const char *type;
        char composedId[OBS_MAX_GRANTEE_USER_ID_SIZE +
            OBS_MAX_GRANTEE_DISPLAY_NAME_SIZE + 16] = { 0 };
        const char *id;
        switch (grant->grantee_type) {
        case OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL:
            type = "Email";
            id = grant->grantee.huawei_customer_by_email.email_address;
            break;
        case OBS_GRANTEE_TYPE_CANONICAL_USER:
            type = "UserID";
            snprintf(composedId, sizeof(composedId),
                "%s (%s)", grant->grantee.canonical_user.id,
                grant->grantee.canonical_user.display_name);
            id = composedId;
            break;
        case OBS_GRANTEE_TYPE_ALL_OBS_USERS:
            type = "Group";
            id = "Authenticated Users";
            break;
        default:
            type = "Group";
            id = "All Users";
            break;
        }
        const char *perm;
        switch (grant->permission) {
        case OBS_PERMISSION_READ:
            perm = "READ";
            break;
        case OBS_PERMISSION_WRITE:
            perm = "WRITE";
            break;
        case OBS_PERMISSION_READ_ACP:
            perm = "READ_ACP";
            break;
        case OBS_PERMISSION_WRITE_ACP:
            perm = "WRITE_ACP";
            break;
        default:
            perm = "FULL_CONTROL";
            break;
        }
        const char *delivered;
        if (grant->bucket_delivered == BUCKET_DELIVERED_FALSE)
            delivered = "false";
        else
            delivered = "true";
        printf("%-6s  %-90s  %-12s  %-8s\n", type, id, perm, delivered);
    }
}
```

相关链接

- 关于获取桶ACL的API说明，请参见**获取桶ACL**。

- 更多关于获取桶ACL的代码示例，请参见**Github示例**。
- 获取桶ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 权限相关常见问题请参见**权限相关常见问题**。

6.7 获取桶存量信息(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

调用获取桶存量信息接口，可查询指定桶内的对象个数，以及对象占用空间的大小。

说明

由于OBS桶存量是后台统计，因此存量会有一定的时延，不能实时更新，因此不建议对存量做实时校验。

接口约束

- 您必须是桶拥有者或拥有获取桶存量信息的权限，才能获取桶存量信息。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketStorage权限，如果使用桶策略则需授予GetBucketStorage权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_storage_info(const obs_options *options, int capacity_length, char *capacity,
                             int object_number_length, char *object_number,
                             obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 6-120 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
capacity_length	int	必选	**参数解释：** 使用容量的缓存大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
capacity	char *	必选	**参数解释：** 使用容量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_number_length	int	必选	**参数解释：** 对象数目的缓存大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_number	char *	必选	**参数解释：** 对象数目的缓存。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-121 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-122 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-123 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 6-124 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-125 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-126 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-127 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-128 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-129 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-130 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-131 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-132 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-133 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORS规则可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-134 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-135 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-136 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶的存量信息

以下示例展示如何获取桶的存量信息：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何获取桶的存量信息：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
```



```
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    //定义桶容量缓存及对象数缓存
    char capacity[OBS_COMMON_LEN_256 + 1] = { 0 };
    char obj_num[OBS_COMMON_LEN_256 + 1] = { 0 };
    // 获取桶存量信息
    get_bucket_storage_info(&options, OBS_COMMON_LEN_256 + 1, capacity, OBS_COMMON_LEN_256 + 1,
obj_num,
        &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("get_bucket_storage_info success,bucket=%s objNum=%s capacity=%s\n",
            bucketName, obj_num, capacity);
    }
    else
    {
        printf("get_bucket_storage_info failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
```



```
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于获取桶存量信息的API说明，请参见**获取桶存量信息**。
- 更多关于获取桶存量信息的代码示例，请参见**Github示例**。
- 获取桶存量信息过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

6.8 设置桶配额(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

桶配额是桶容量的上限值。默认情况下，OBS系统和单个桶都没有总数据容量和对象数量的限制。您可以根据需要，为桶设置配额限制来控制桶内允许上传的对象总容量，超过设置的对象容量后，上传对象会失败。例如您为桶设置配额为100G，那么当桶内所有对象的大小总和达到100G后，再上传对象就会因为超过配额导致上传失败。

容量限制只对桶配额设置生效后的对象上传操作有影响， 如果您设置桶配额时，桶配额的数值小于桶中已上传的对象容量，并不会导致桶中已有对象被删除，但该桶后续将不能再上传任何对象。这种情况下只有删除部分已有对象，将已用空间释放到配额限制以下才能再次上传新对象。

接口约束

- 桶配额值必须为非负整数，单位为字节，支持的最大值为2^{63}-1。
- OBS没有提供删除桶配额的接口，您可以将桶配额设置为0来取消配额限制。
- 您必须是桶拥有者或拥有设置桶配额的权限，才能设置桶配额。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketQuota权限，如果使用桶策略则需授予PutBucketQuota权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_quota(const obs_options *options, uint64_t storage_quota,
                      obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 6-137 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
storage_quota	uint64_t	必选	**参数解释：** 配额大小，单位字节。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-138 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-139 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-140 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-141 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-142 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-143 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-144 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-145 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-146 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-147 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-148 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-149 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-150 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 长度为32的字符串。 **取值范围：** 无 **默认取值**： 无
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-151 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-152 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-153 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。

枚举值	说明
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置桶配额

以下示例展示如何设置桶配额：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置桶配额：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    uint64_t bucketquota = 100 * 1024 * 1024 * 1024;
    // 设置桶配额
    set_bucket_quota(&options, bucketquota, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("set bucket quota successfully. \n");
    }
    else
    {
        printf("set bucket quota failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
```



```
    }
  }
// 打印响应信息
#define print_nonnull(name, field)                              \
  do {                                                          \
    if (properties-> field) {                                   \
      printf("%s: %s\n", name, properties->field);          \
    }                                                           \
  } while (0)
  print_nonnull("request_id", request_id);
  print_nonnull("request_id2", request_id2);
  print_nonnull("content_type", content_type);
  if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
  }
  print_nonnull("server", server);
  print_nonnull("ETag", etag);
  print_nonnull("expiration", expiration);
  print_nonnull("website_redirect_location", website_redirect_location);
  print_nonnull("version_id", version_id);
  print_nonnull("allow_origin", allow_origin);
  print_nonnull("allow_headers", allow_headers);
  print_nonnull("max_age", max_age);
  print_nonnull("allow_methods", allow_methods);
  print_nonnull("expose_headers", expose_headers);
  print_nonnull("storage_class", storage_class);
  print_nonnull("server_side_encryption", server_side_encryption);
  print_nonnull("kms_key_id", kms_key_id);
  print_nonnull("customer_algorithm", customer_algorithm);
  print_nonnull("customer_key_md5", customer_key_md5);
  print_nonnull("bucket_location", bucket_location);
  print_nonnull("obs_version", obs_version);
  print_nonnull("restore", restore);
  print_nonnull("obs_object_type", obs_object_type);
  print_nonnull("obs_next_append_position", obs_next_append_position);
  print_nonnull("obs_head_epid", obs_head_epid);
  print_nonnull("reserved_indicator", reserved_indicator);
  int i;
  for (i = 0; i < properties->meta_data_count; i++) {
    printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
      properties->meta_data[i].value);
  }
  return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回
调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
  if (callback_data) {
    obs_status *ret_status = (obs_status *)callback_data;
    *ret_status = status;
  } else {
    printf("Callback_data is NULL");
  }
  if (error && error->message) {
    printf("Error Message: \n   %s\n", error->message);
  }
  if (error && error->resource) {
    printf("Error Resource: \n  %s\n", error->resource);
  }
  if (error && error->further_details) {
    printf("Error further_details: \n   %s\n", error->further_details);
  }
  if (error && error->extra_details_count) {
    int i;
    for (i = 0; i < error->extra_details_count; i++) {
      printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
          error->extra_details[i].value);
    }
```



```
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶配额的API说明，请参见**设置桶配额**。
- 更多关于设置桶配额的代码示例，请参见**Github示例**。
- 设置桶配额过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

6.9 获取桶配额(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

桶配额是桶容量的上限值。调用获取桶配额接口，可获取指定桶的配额值。桶配额为0代表桶容量没有上限。

接口约束

- 桶配额值必须为非负整数，单位为字节，支持的最大值为2^{63}-1。
- 桶拥有者的状态是欠费冻结时不可以查询桶配额信息。
- 您必须是桶拥有者或拥有获取桶配额的权限，才能获取桶配额。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketQuota权限，如果使用桶策略则需授予GetBucketQuota权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_quota(const obs_options *options, uint64_t *storagequota_return,
                      obs_response_handler *handler,  void *callback_data);
```

请求参数说明

表 6-154 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
storagequota_return	uint64_t *	必选	**参数解释：** 获取到的配额大小，单位字节。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-155 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-156 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-157 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-158 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-159 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-160 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 6-161 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-162 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-163 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_callback	void(*temp_auth_callback)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-164 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-165 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-166 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 6-167 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-168 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-169 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-170 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶配额

您可以通过函数get_bucket_quota获取桶配额：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何获取桶配额：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
```



```
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    // 获取桶配额
    uint64_t bucketquota = 0;
    get_bucket_quota(&options, &bucketquota, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("Bucket=%s  Quota=%lu \n get bucket quota successfully. \n ",
            bucketName, bucketquota);
    }
    else
    {
        printf("get bucket quota failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                  \
        if (properties-> field) {                             \
            printf("%s: %s\n", name, properties->field);          \
        }                                                 \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
```



```
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于获取桶配额的API说明，请参见**获取桶配额**。
- 更多关于获取桶配额的代码示例，请参见**Github示例**。
- 获取桶配额过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

6.10 设置桶存储类别(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS提供了这些存储类型：标准存储、低频访问存储、归档存储、深度归档存储（受限公测中），从而满足客户业务对存储性能、成本的不同诉求，详情可参见OBS**存储类别**。

调用设置桶存储类别接口，可设置指定桶的存储类别。设置了桶的默认存储类别之后，如果上传对象、复制对象和初始化多段上传任务时未指定对象的存储类别，则该对象的存储类别默认与桶的存储类别保持一致。

接口约束

- 您必须是桶拥有者或拥有设置桶存储类别的权限，才能设置桶存储类别。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:PutBucketStoragePolicy权限，如果使用桶策略则需授予PutBucketStoragePolicy权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_storage_class_policy(const obs_options *options,
        obs_storage_class storage_class_policy, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 6-171 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
storage_class_policy	**obs_storage_class**	必选	**参数解释：** 桶存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-172 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-173 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-174 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-175 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-176 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-177 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-178 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-179 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-180 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-181 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-182 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-183 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties** *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-184 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-185 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 长度为32的字符串。 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-186 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-187 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-188 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置桶存储类别

以下示例展示如何设置桶存储类别：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置桶存储类别：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
```

```
    obs_status  ret_status = OBS_STATUS_BUTT;
    obs_storage_class storage_class_policy = OBS_STORAGE_CLASS_GLACIER;
    set_bucket_storage_class_policy(&options, storage_class_policy,
        &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set bucket storage class successfully.");
    }
    else
    {
        printf("set bucket storage class failed(%s).\n",
            obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                  \
        if (properties-> field) {                             \
            printf("%s: %s\n", name, properties->field);          \
        }                                                 \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
```



```
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶存储类别的API说明，请参见**设置桶存储类型**。
- 更多关于设置桶存储类别的代码示例，请参见**Github示例**。
- 设置桶存储类别过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

6.11 获取桶存储类别(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS提供了这些存储类型：标准存储、低频访问存储、归档存储、深度归档存储（受限公测中），从而满足客户业务对存储性能、成本的不同诉求，详情可参见OBS**存储类别**。

调用获取桶存储类别接口，可获取指定桶的存储类别。

接口约束

- 您必须是桶拥有者或拥有获取桶存储类别的权限，才能获取桶存储类别。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketStoragePolicy权限，如果使用桶策略则需授予GetBucketStoragePolicy权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_storage_class_policy(const obs_options *options,
                                     obs_get_bucket_storage_class_handler *handler, void *callback_data);
```

请求参数说明

表 6-189 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_get_bucket_storage_class_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-190 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 6-191 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 6-192 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 6-193 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 6-194 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 6-195 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 6-196 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-197 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-198 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-199 obs_get_bucket_storage_class_handler

参数名称	参数类型	是否必选	描述
response_handler	obs_response_handler *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
get_bucket_storage_class_callback	obs_get_bucket_storage_policy_callback *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无 **取值范围：** 请详见obs_get_bucket_storage_policy_callback。

表 6-200 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 6-201 obs_get_bucket_storage_policy_callback

参数名称	参数类型	是否必选	描述
storage_class_policy	const char *	必选	**参数解释：** 桶的存储类别。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-202 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-203 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 6-204 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 长度为32的字符串。 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-205 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-206 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 6-207 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶存储类别

以下示例展示如何获取桶存储类别：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status get_bucket_storageclass_handler(const char * storage_class, void * callBackData);
int main()
{
    // 以下示例展示如何获取桶存储类别：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置响应回调函数
    obs_get_bucket_storage_class_handler getBucketStorageResponse =
    {
        {response_properties_callback, &response_complete_callback},
        &get_bucket_storageclass_handler
```



```
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    //获取桶存储类别
    get_bucket_storage_class_policy(&options, &getBucketStorageResponse, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("get bucket storage class successfully.\n");
    }
    else
    {
        printf("get bucket storage class failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
```



```
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
obs_status get_bucket_storageclass_handler(const char * storage_class, void * callBackData)
{
    printf("Bucket storage class is: %s\n", storage_class);
    return OBS_STATUS_OK;
}
```

相关链接

- 关于获取桶存储类别的API说明，请参见**获取桶存储类型**。
- 更多关于获取桶存储类别的代码示例，请参见**Github示例**。
- 获取桶存储类别过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

7 上传对象(C SDK)

7.1 流式上传(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以将本地文件直接通过Internet上传至OBS指定的位置。待上传的文件可以是任何类型：文本文件、图片、视频等。

通过SDK的流式上传，可以上传小于5GB的文件。本章节介绍如何使用C SDK流式上传对象。

可以通过put_object上传数据流到OBS。

接口约束

- 您必须是桶拥有者或拥有上传对象的权限，才能上传对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 单次上传对象大小范围是[0, 5GB]。
- 如果需要上传超过5GB的大文件，需要通过**多段操作**来分段上传。

方法定义

```
void put_object(const obs_options *options, char *key, uint64_t content_length,
                obs_put_properties *put_properties,
                server_side_encryption_params *encryption_params,
                obs_put_object_handler *handler, void *callback_data);
```

请求参数说明

表 7-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。
put_properties	**obs_put_properties***	必选	**参数解释：** 上传对象属性。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_put_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 7-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 7-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-5 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-6 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-7 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-8 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-11 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（指从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。 **默认取值：** 无
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD %E6%96%87: %E4%B8%AD %E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。 **默认取值：** 无
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-12 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** **0（指从对象的第一个字节开始下载）**
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-13 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-14 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-15 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。

枚举值	说明
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-16 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。 **约束限制：** 无 **取值范围：** • application/x-www-form-urlencoded • application/json **默认取值：** 如果不设置取值，则默认为application/json。

表 7-17 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-18 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-19 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-20 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-21 obs_put_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
put_object_data_callback	**obs_put_object_data_callback** *	必选	**参数解释:** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制:** 无
progress_callback	**obs_progress_callback_internal** *	必选	**参数解释:** 进度回调函数指针。 **约束限制:** 无

表 7-22 obs_put_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释:** buffer的长度。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
buffer	char *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-23 obs_progress_callback_internal

参数名称	参数类型	是否必选	描述
now	uint64_t	必选	**参数解释：** 已上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
total	uint64_t	必选	**参数解释：** 总共需要上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-24 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-25 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-26 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-27 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-28 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-29 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-30 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：流式上传

以下示例展示如何通过put_object流式上传对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct put_buffer_object_callback_data
{
    char *put_buffer;
    uint64_t buffer_size;
    uint64_t cur_offset;
    obs_status ret_status;
} put_buffer_object_callback_data;
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data);
char* generate_upload_buffer(uint64_t buffer_size);
int main()
{
    // 以下示例展示如何通过put_object流式上传对象
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
```

```
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //初始化存储上传数据的结构体
    put_buffer_object_callback_data data;
    memset(&data, 0, sizeof(put_buffer_object_callback_data));
    // 设置buffersize
    data.buffer_size = 10 * 1024 * 1024;
    // 创建模拟的流式上传数据buffer, 并赋值到上传数据结构中
    data.put_buffer = generate_upload_buffer(data.buffer_size);
    if (NULL == data.put_buffer) {
        printf("generate put buffer failed. \n");
        return;
    }
    // 上传的对象名
    char *key = "example_put_buffer_test.txt";
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_buffer_complete_callback },
          &put_buffer_data_callback
    };
    put_object(&options, key, data.buffer_size, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from buffer successfully. \n");
    }
    else
    {
        printf("put object from buffer failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    // 释放模拟的流式上传数据buffer
    free(data.put_buffer);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
```



```
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        put_buffer_object_callback_data *data = (put_buffer_object_callback_data *)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data)
{
    put_buffer_object_callback_data *data =
        (put_buffer_object_callback_data *)callback_data;
    int toRead = 0;
    if (data->buffer_size) {
        toRead = ((data->buffer_size > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->buffer_size);
        memcpy_s(buffer, buffer_size, data->put_buffer + data->cur_offset, toRead);
    }
```



```
    uint64_t originalContentLength = data->buffer_size;
    data->buffer_size -= toRead;
    data->cur_offset += toRead;
    if (data->buffer_size) {
        printf("%llu bytes remaining ", (unsigned long long)data->buffer_size);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->buffer_size) * 100) / originalContentLength));
    }
    return toRead;
}
// 创建模拟的流式上传数据
char* generate_upload_buffer(uint64_t buffer_size) {
    void* upload_buffer = NULL;
    if (buffer_size > 0) {
        upload_buffer = malloc(buffer_size);
        if (upload_buffer != NULL) {
            memset(upload_buffer, 't', buffer_size);
        }
    }
    return upload_buffer;
}
```

相关链接

- 关于上传对象-PUT上传的API说明，请参见**PUT上传**。
- 更多关于上传对象的代码示例，请参见**Github示例**。
- 上传对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 上传对象常见问题请参见**上传对象失败**。

7.2 文件上传(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以将本地文件直接通过Internet上传至OBS指定的桶中。待上传的文件可以是任何类型：文本文件、图片、视频等。

本章节介绍如何使用文本上传对象。

接口约束

- 您必须是桶拥有者或拥有上传对象的权限，才能上传对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 单次上传对象大小范围是[0, 5GB]。
- 如果需要上传超过5GB的大文件，需要通过**多段操作**来分段上传。

方法定义

```
void put_object(const obs_options *options, char *key, uint64_t content_length,
                obs_put_properties *put_properties,
                server_side_encryption_params *encryption_params,
                obs_put_object_handler *handler, void *callback_data);
```

请求参数说明

表 7-31 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。

参数名称	参数类型	是否必选	描述
put_properties	obs_put_properties*	必选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_put_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-32 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 7-33 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 7-34 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-35 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-36 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-37 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-38 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-39 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-40 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-41 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（指从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。 **默认取值：** 无
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD %E6%96%87: %E4%B8%AD %E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。 **默认取值：** 无
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-42 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** **0（指从对象的第一个字节开始下载）**
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-43 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-44 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-45 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。

枚举值	说明
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-46 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。 **约束限制：** 无 **取值范围：** • application/x-www-form-urlencoded • application/json **默认取值：** 如果不设置取值，则默认为application/json。

表 7-47 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-48 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-49 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-50 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-51 obs_put_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
put_object_data_callback	**obs_put_object_data_callback** *	必选	**参数解释:** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制:** 无
progress_callback	**obs_progress_callback_internal** *	必选	**参数解释:** 进度回调函数指针。 **约束限制:** 无

表 7-52 obs_put_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释:** buffer的长度。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
buffer	char *	必选	**参数解释:** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-53 obs_progress_callback_internal

参数名称	参数类型	是否必选	描述
now	uint64_t	必选	**参数解释:** 已上传的字节数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
total	uint64_t	必选	**参数解释:** 总共需要上传的字节数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-54 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-55 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-56 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-57 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-58 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-59 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-60 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：文件上传

以下示例展示如何通过put_object上传文件：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何通过put_object上传文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
```

```
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_test";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
```



```
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
```



```
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

说明

- 上传的内容大小不能超过5GB。
- 上传文件流时，必须以二进制模式（“rb”模式或者“rb+”模式）打开文件。

相关链接

- 关于上传对象-POST上传的API说明，请参见**POST上传**。
- 更多关于上传对象的代码示例，请参见**Github示例**。
- 上传对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 上传对象常见问题请参见**上传对象失败**。

7.3 创建文件夹(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以在已创建的桶中新建一个文件夹，从而更方便的对存储在OBS中的数据进行分类管理。

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。创建文件夹实际上是创建了一个大小为0且对象名以“/”结尾的对象，这类对象与其他对象无任何差异，可以进行下载、删除等操作，只是OBS控制台会将这类以“/”结尾的对象以文件夹的方式展示。

接口约束

- 您必须是桶拥有者或拥有创建文件夹的权限，才能创建文件夹。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 创建文件夹本质上来说是创建了一个大小为0且对象名以“/”结尾的对象。
- 多级文件夹创建最后一级即可，比如src1/src2/src3/，创建src1/src2/src3/即可，无需创建src1/、src1/src2/。

方法定义

```
void put_object(const obs_options *options, char *key, uint64_t content_length,
                obs_put_properties *put_properties,
                server_side_encryption_params *encryption_params,
                obs_put_object_handler *handler, void *callback_data);
```

请求参数说明

表 7-61 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。

参数名称	参数类型	是否必选	描述
put_properties	obs_put_properties*	必选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_put_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-62 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 7-63 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 7-64 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-65 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-66 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-67 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-68 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-69 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-70 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-71 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（指从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。 **默认取值：** 无
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD %E6%96%87: %E4%B8%AD %E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。 **默认取值：** 无
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-72 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** **0（指从对象的第一个字节开始下载）**
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-73 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-74 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-75 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。

枚举值	说明
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-76 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。 **约束限制：** 无 **取值范围：** • application/x-www-form-urlencoded • application/json **默认取值：** 如果不设置取值，则默认为application/json。

表 7-77 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见 image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-78 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-79 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-80 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-81 obs_put_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
put_object_data_callback	**obs_put_object_data_callback** *	必选	**参数解释:** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制:** 无
progress_callback	**obs_progress_callback_internal** *	必选	**参数解释:** 进度回调函数指针。 **约束限制:** 无

表 7-82 obs_put_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释:** buffer的长度。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
buffer	char *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-83 obs_progress_callback_internal

参数名称	参数类型	是否必选	描述
now	uint64_t	必选	**参数解释：** 已上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
total	uint64_t	必选	**参数解释：** 总共需要上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-84 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-85 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-86 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-87 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-88 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-89 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-90 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：创建文件夹

以下示例展示如何创建文件夹（本质上来说是创建了一个大小为0且对象名以“/”结尾的对象）：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何创建文件夹（本质上来说是创建了一个大小为0且对象名以“/”结尾的对象）：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
```



```
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_folder/";
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    put_object(&options, key, 0, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
```



```
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
```

相关链接

- 通过API创建文件夹，本质上来说是创建了一个大小为0且对象名以“/”结尾的对象。关于创建文件夹的API说明，请参见**PUT上传**。
- 创建文件夹过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

7.4 断点续传上传(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

对分段上传的封装和加强，解决上传大文件时由于网络不稳定或程序崩溃导致上传失败的问题。其原理是将待上传的文件分成若干个分段分别上传，并实时地将每段上传结果统一记录在checkpoint文件中，仅当所有分段都上传成功时返回上传成功的结果，否则返回错误信息提醒用户再次调用接口进行重新上传（重新上传时因为有checkpoint文件记录当前的上传进度，避免重新上传所有分段，从而节省资源提高效率）。

您可以通过upload_file进行断点续传上传。

接口约束

- 您必须是桶拥有者或拥有上传对象的权限，才能上传对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 断点续传上传接口传入的文件大小至少要100K。
- 使用SDK的断点续传接口时，必须开启断点续传选项后才能在进程再次进入时读取上一次上传的进度。

方法定义

```
void upload_file(const obs_options *options, char *key, server_side_encryption_params *encryption_params,
    obs_upload_file_configuration *upload_file_config,
    obs_upload_file_server_callback server_callback,
    obs_upload_file_response_handler *handler,
    void *callback_data);
```

请求参数说明

表 7-91 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
upload_file_config	obs_upload_file_configuration *	必选	**参数解释：** 上传文件的配置。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_upload_file_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-92 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无

表 7-93 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可选择的访问策略选项参见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 7-94 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-95 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-96 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-97 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-98 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-99 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-100 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-101 obs_upload_file_configuration

参数名称	参数类型	是否必选	描述
upload_file	char *	必选	**参数解释：** 要上传的本地文件路径。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
part_size	uint64_t	可选	**参数解释：** 分段大小，单位字节。 **约束限制：** 无 **取值范围：** 100KB~5GB **默认取值：** 5MB
check_point_file	char *	可选	**参数解释：** 记录上传进度的文件，只在断点续传模式下有效。当该值为空时，默认与待上传的本地文件同目录。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
enable_check_point	int	可选	**参数解释：** 是否启用断点续传。 **约束限制：** 无 **取值范围：** ● 0（表示不启用断点续传模式，此时断点续传上传接口退化成对分段上传的简单封装，不会产生checkpoint文件） ● 1（表示启用断点续传模式） **默认取值：** 0（表示不开启）
task_num	int	可选	**参数解释：** 分段上传时的最大并发数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 1

参数名称	参数类型	是否必选	描述
pause_upload_flag	int *	可选	**参数解释：** 暂停上传标志。 **约束限制：** 不能为NULL，指向的值为1时暂停上传。 **取值范围：** 无 **默认取值：** 无
put_properties	**obs_put_properties** *	必选	**参数解释：** 对象相关属性。 **约束限制：** 无

表 7-102 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-103 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-104 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-105 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-106 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-107 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-108 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-109 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-110 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-111 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-112 obs_upload_file_response_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
upload_file_callback	**obs_upload_file_callback** *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
progress_callback	**obs_progress_callback** *	必选	**参数解释：** 进度回调函数指针。 **约束限制：** 无

表 7-113 obs_upload_file_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** 请求状态。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。

参数名称	参数类型	是否必选	描述
result_message	char *	必选	**参数解释:** 上传结果。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
part_count_return	int	必选	**参数解释:** 段数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
upload_info_list	**obs_upload_file_part_info** *	必选	**参数解释:** 段信息。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-114 obs_upload_file_part_info

参数名称	参数类型	是否必选	描述
part_num	**obs_response_handler** *	必选	**参数解释:** 分段号。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	必选	**参数解释：** 分段起始位置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
part_size	uint64_t	必选	**参数解释：** 分段大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
status_return	part_upload_status	必选	**参数解释：** 分段状态。 **约束限制：** 无 **取值范围：** 请详见part_upload_status。

表 7-115 part_upload_status

枚举值	说明
UPLOAD_NOTSTART	上传未开始。
UPLOADING	上传中。
UPLOAD_FAILED	上传失败。
UPLOAD_SUCCESS	上传成功。
STATUS_BUTT	默认的状态。

表 7-116 obs_progress_callback

参数名称	参数类型	是否必选	描述
progress	double	必选	**参数解释:** 进度百分比。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
uploadedSize	uint64_t	必选	**参数解释:** 已上传的字节数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
fileTotalSize	uint64_t	必选	**参数解释:** 总共需要上传的字节数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-117 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-118 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties** *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-119 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-120 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 长度为32的字符串。 **取值范围：** 无 **默认取值**： 无
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	const obs_name_value *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-121 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
error_headers	char**	**参数解释:** 响应头域中包含error的所有头域。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-122 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
value	char *	可选	**参数解释:** 属性的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-123 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。

枚举值	说明
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：断点续传上传

以下示例展示如何使用断点续传上传接口上传文件：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
void uploadFileResultCallback(obs_status status,
    char *resultMsg,
    int partCountReturn,
    obs_upload_file_part_info * uploadInfoList,
    void *callbackData);
void test_progress_callback(double progress, uint64_t uploadedSize, uint64_t fileTotalSize, void *callback_data);
int main()
{
    // 以下示例展示如何使用断点续传上传接口上传文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    //初始化结构体put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    obs_upload_file_configuration uploadFileInfo;
    memset(&uploadFileInfo, 0, sizeof(obs_upload_file_configuration));
    uploadFileInfo.check_point_file = 0;
    uploadFileInfo.enable_check_point = 1;
    uploadFileInfo.part_size = 5L * 1024 * 1024;
    uploadFileInfo.task_num = 8;
    uploadFileInfo.upload_file = "./example_local_file_to_upload.tar";
    uploadFileInfo.put_properties = &put_properties;
    int pause_upload_flag = 0;
    uploadFileInfo.pause_upload_flag = &pause_upload_flag;
    // 断点续传上传对象名
    char *key = "example_upload_file_object_key";
    obs_status ret_status = OBS_STATUS_BUTT;
    //回调函数
    obs_upload_file_response_handler Handler =
    {
        {&response_properties_callback, &response_complete_callback},
        &uploadFileResultCallback, &test_progress_callback
    };
    obs_upload_file_server_callback server_callback;
    init_server_callback(&server_callback);
    initialize_break_point_lock();
    upload_file(&options, key, 0, &uploadFileInfo, server_callback, &Handler, &ret_status);
    deinitialize_break_point_lock();
    if (OBS_STATUS_OK == ret_status) {
        printf("test upload file successfully. \n");
    }
    else
```



```
    {
        printf("test upload file faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
```



```
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
//uploadFileResultCallback示例，printf语句可以替换为自定义的日志打印语句
void uploadFileResultCallback(obs_status status,
    char *resultMsg,
    int partCountReturn,
    obs_upload_file_part_info * uploadInfoList,
    void *callbackData)
{
    int i = 0;
    obs_upload_file_part_info * pstUploadInfoList = uploadInfoList;
    printf("status return is %d(%s)\n", status, obs_get_status_name(status));
    printf("%s", resultMsg);
    printf("partCount[%d]\n", partCountReturn);
    for (i = 0; i < partCountReturn; i++)
    {
        printf("partNum[%d],startByte[%llu],partSize[%llu],status[%d]\n",
            pstUploadInfoList->part_num,
            pstUploadInfoList->start_byte,
            pstUploadInfoList->part_size,
            pstUploadInfoList->status_return);
        pstUploadInfoList++;
    }
    if (callbackData) {
        obs_status* retStatus = (obs_status*)callbackData;
        (*retStatus) = status;
    }
}
static double g_progress = 0;
void test_progress_callback(double progress, uint64_t uploadedSize, uint64_t fileTotalSize, void
*callback_data) {
    if (progress == 100 || (g_progress < progress && progress - g_progress > 2)) {
        printf("test_progress_callback progress=%f  uploadedSize=%llu fileTotalSize=%llu  callback_data=%p
\n", progress, uploadedSize, fileTotalSize, callback_data);
        g_progress = progress;
    }
}
```


相关链接

- 更多关于上传对象的代码示例，请参见**Github示例**。
- 上传对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 上传对象常见问题请参见**上传对象失败**。

7.5 追加写对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以根据需要上传文件或者文件夹至已有的OBS桶。待上传的文件可以是任何类型：文本文件、图片、视频等。

追加写对象操作是指在指定对象尾追加上传数据，不存在相同对象键值的对象则创建新对象。

每次追加上传都会更新该对象的最后修改时间。

接口约束

- 您必须是桶拥有者或拥有上传对象的权限，才能上传对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 启用服务端加密SSE-C方式时，追加上传和初始化段一样，设置诸如x-obs-server-side-encryption之类的请求Header，后续追加上传也必须携带。
- 启用服务端加密SSE-KMS方式时，有且只有第一次上传且桶内不存在同名对象时，才设置诸如x-obs-server-side-encryption之类的请求Header，后续追加上传不携带，会继承之前的设置。
- 每次追加上传的长度不能超过对象长度上限5G的限制。
- 每个Appendable对象追加上传次数最多为10000次。
- 如果对象存储类别为COLD（归档存储）或DEEP_ARCHIVE（深度归档存储），则不能调用该接口。
- 如果桶设置了跨区域复制配置，则不能调用该接口。
- 并行文件系统不支持追加上传对象。
- put_object上传的对象可覆盖append_object上传的对象，覆盖后对象变为普通对象，不可再进行追加上传。
- 第一次调用追加上传时，如果已存在同名的普通对象，则会抛出异常（HTTP状态码为409）。
- 追加上传返回的ETag是当次追加数据内容的ETag，不是完整对象的ETag。

方法定义

```
void append_object(const obs_options *options, char *key, uint64_t content_length, const char * position,
                   obs_put_properties *put_properties,server_side_encryption_params *encryption_params,
                   obs_append_object_handler *handler, void *callback_data);
```

请求参数

表 7-124 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。

参数名称	参数类型	是否必选	描述
position	char *	必选	**参数解释：** 追加写的起始位置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
put_properties	**obs_put_properties***	必选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	**obs_append_object_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-125 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 7-126 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 7-127 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-128 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-129 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-130 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-131 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-132 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-133 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-134 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87:%E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-135 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-136 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-137 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
value	char *	可选	**参数解释:** 属性的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-138 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释:** 指定下载对象的起始位置。 **取值范围:** [0~对象长度-1)，单位：字节。 **默认取值:** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释:** 指定下载的长度。 **取值范围:** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值:** 无

参数名称	参数类型	是否必选	描述
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-139 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。

枚举值	说明
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-140 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-141 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-142 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-143 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-144 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-145 obs_append_object_handler

参数名称	参数类型	是否必选	描述
response_handler	obs_response_handler *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
append_object_data_callback	obs_append_object_data_callback *	必选	**参数解释：** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制：** 无

表 7-146 obs_append_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-147 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-148 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-149 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-150 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** 请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-151 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-152 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。

枚举值	说明
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：追加写对象

以下示例展示如何通过append_object实现追加上传：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct put_buffer_object_callback_data
{
    char *put_buffer;
    uint64_t buffer_size;
    uint64_t cur_offset;
    obs_status ret_status;
} put_buffer_object_callback_data;
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data);
char* generate_upload_buffer(uint64_t buffer_size);
int main()
```



```
{
    // 以下示例展示如何通过append_object实现追加上传：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //初始化存储上传数据的结构体
    put_buffer_object_callback_data data;
    memset(&data, 0, sizeof(put_buffer_object_callback_data));
    // 设置buffersize
    data.buffer_size = 10 * 1024 * 1024;
    // 创建模拟的流式上传数据buffer, 并赋值到上传数据结构中
    data.put_buffer = generate_upload_buffer(data.buffer_size);
    if (NULL == data.put_buffer) {
        printf("generate put buffer failed. \n");
        return;
    }
    // 上传的对象名
    char *key = "example_put_file_test_append";
    // append上传的起始位置
    char *position = "0";
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_buffer_complete_callback },
          &put_buffer_data_callback
    };
    append_object(&options, key, data.buffer_size, position, &put_properties,
        0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("append object from buffer successfully. \n");
    }
    else
    {
        printf("append object from buffer failed(%s).\n", obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
```



```
        return OBS_STATUS_OK;
    }
  }
  // 打印响应信息
#define print_nonnull(name, field)                              \
  do {                                                          \
    if (properties-> field) {                                   \
      printf("%s: %s\n", name, properties->field);          \
    }                                                           \
  } while (0)
  print_nonnull("request_id", request_id);
  print_nonnull("request_id2", request_id2);
  print_nonnull("content_type", content_type);
  if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
  }
  print_nonnull("server", server);
  print_nonnull("ETag", etag);
  print_nonnull("expiration", expiration);
  print_nonnull("website_redirect_location", website_redirect_location);
  print_nonnull("version_id", version_id);
  print_nonnull("allow_origin", allow_origin);
  print_nonnull("allow_headers", allow_headers);
  print_nonnull("max_age", max_age);
  print_nonnull("allow_methods", allow_methods);
  print_nonnull("expose_headers", expose_headers);
  print_nonnull("storage_class", storage_class);
  print_nonnull("server_side_encryption", server_side_encryption);
  print_nonnull("kms_key_id", kms_key_id);
  print_nonnull("customer_algorithm", customer_algorithm);
  print_nonnull("customer_key_md5", customer_key_md5);
  print_nonnull("bucket_location", bucket_location);
  print_nonnull("obs_version", obs_version);
  print_nonnull("restore", restore);
  print_nonnull("obs_object_type", obs_object_type);
  print_nonnull("obs_next_append_position", obs_next_append_position);
  print_nonnull("obs_head_epid", obs_head_epid);
  print_nonnull("reserved_indicator", reserved_indicator);
  int i;
  for (i = 0; i < properties->meta_data_count; i++) {
    printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
        properties->meta_data[i].value);
  }
  return OBS_STATUS_OK;
}
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
  if (callback_data) {
    put_buffer_object_callback_data *data = (put_buffer_object_callback_data *)callback_data;
    data->ret_status = status;
  }
  else {
    printf("Callback_data is NULL");
  }
  if (error && error->message) {
    printf("Error Message: \n   %s\n", error->message);
  }
  if (error && error->resource) {
    printf("Error Resource: \n  %s\n", error->resource);
  }
  if (error && error->further_details) {
    printf("Error further_details: \n   %s\n", error->further_details);
  }
  if (error && error->extra_details_count) {
    int i;
    for (i = 0; i < error->extra_details_count; i++) {
      printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
          error->extra_details[i].value);
    }
```



```
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data)
{
    put_buffer_object_callback_data *data =
        (put_buffer_object_callback_data *)callback_data;
    int toRead = 0;
    if (data->buffer_size) {
        toRead = ((data->buffer_size > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->buffer_size);
        memcpy_s(buffer, buffer_size, data->put_buffer + data->cur_offset, toRead);
    }
    uint64_t originalContentLength = data->buffer_size;
    data->buffer_size -= toRead;
    data->cur_offset += toRead;
    if (data->buffer_size) {
        printf("%llu bytes remaining ", (unsigned long long)data->buffer_size);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->buffer_size) * 100) / originalContentLength));
    }
    return toRead;
}
// 创建模拟的流式上传数据
char* generate_upload_buffer(uint64_t buffer_size) {
    void* upload_buffer = NULL;
    if (buffer_size > 0) {
        upload_buffer = malloc(buffer_size);
        if (upload_buffer != NULL) {
            memset(upload_buffer, 't', buffer_size);
        }
    }
    return upload_buffer;
}
```

相关链接

- 关于追加写的API说明，请参见**追加写对象**。
- 更多关于追加写的代码示例，请参见**Github示例**。
- 追加写过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**上传对象失败**。

7.6 修改写对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

将指定并行文件系统内的一个对象从指定位置起修改为其他内容。

接口约束

- 目前接口仅在并行文件系统支持，普通对象桶不支持，如何创建并行文件系统请参考**创建桶**。
- 您必须是并行文件系统拥有者或拥有修改写对象的权限，才能修改写对象。建议使用IAM或策略进行授权，如果使用IAM则需授予obs:bucket:PutObject权限，如果使用策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 第一次调用该接口时，如果修改的对象不存在，则会报错（HTTP状态码为404）。

方法定义

```
void modify_object(const obs_options *options, char *key, uint64_t content_length, uint64_t position,
                   obs_put_properties *put_properties,server_side_encryption_params *encryption_params,
                   obs_modify_object_handler *handler, void *callback_data);
```

请求参数说明

表 7-153 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。
position	char *	必选	**参数解释：** 追加写的起始位置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
put_properties	**obs_put_properties***	必选	**参数解释：** 上传对象属性。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	obs_modify_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-154 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 7-155 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 7-156 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 7-157 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 7-158 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 7-159 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 7-160 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-161 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数计计算签名名** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 7-162 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-163 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见obs_canned_acl。
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 7-164 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 7-165 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 7-166 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
value	char *	可选	**参数解释:** 属性的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-167 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释:** 指定下载对象的起始位置。 **约束限制:** 无 **取值范围:** [0~对象长度-1)，单位：字节。 **默认取值:** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 7-168 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 7-169 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为 application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-170 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释:** 图片处理参数头。 **约束限制:** 无 **取值范围:** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释:** 图片处理相关参数。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 7-171 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 7-172 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释:** 加密类型。 **约束限制:** 无 **取值范围:** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 7-173 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 7-174 obs_modify_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
modify_object_data_callback	**obs_modify_object_data_callback** *	必选	**参数解释:** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制:** 无

表 7-175 obs_modify_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释:** buffer的长度。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
buffer	char *	必选	**参数解释:** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-176 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 7-177 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties** *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-178 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 7-179 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-180 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 7-181 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。

枚举值	说明
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：修改写对象

以下示例展示通过modify_object接口修改写对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct put_buffer_object_callback_data
{
    char *put_buffer;
    uint64_t buffer_size;
    uint64_t cur_offset;
    obs_status ret_status;
} put_buffer_object_callback_data;
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data);
char* generate_upload_buffer(uint64_t buffer_size);
int main()
```



```
{
    // 以下示例展示如何通过modify_object接口修改写（只适用于并行文件系统的桶）：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "example-posix-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //初始化存储上传数据的结构体
    put_buffer_object_callback_data data;
    memset(&data, 0, sizeof(put_buffer_object_callback_data));
    // 设置buffersize
    data.buffer_size = 10 * 1024 * 1024;
    // 创建模拟的流式上传数据buffer, 并赋值到上传数据结构中
    data.put_buffer = generate_upload_buffer(data.buffer_size);
    if (NULL == data.put_buffer) {
        printf("generate put buffer failed. \n");
        return;
    }
    // 上传的对象名
    char *key = "example_put_file_test_modify";
    // modify的起始位置
    char *position = "0";
    obs_put_object_handler modify0bjectHandler =
    {
        { &response_properties_callback, &put_buffer_complete_callback },
          &put_buffer_data_callback
    };
    modify_object(&options, key, data.buffer_size, position, &put_properties,
        0, &modify0bjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("modify object from buffer successfully. \n");
    }
    else
    {
        printf("modify object from buffer failed(%s).\n", obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
```



```
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        put_buffer_object_callback_data *data = (put_buffer_object_callback_data *)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
```



```
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data)
{
    put_buffer_object_callback_data *data =
        (put_buffer_object_callback_data *)callback_data;
    int toRead = 0;
    if (data->buffer_size) {
        toRead = ((data->buffer_size > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->buffer_size);
        memcpy_s(buffer, buffer_size, data->put_buffer + data->cur_offset, toRead);
    }
    uint64_t originalContentLength = data->buffer_size;
    data->buffer_size -= toRead;
    data->cur_offset += toRead;
    if (data->buffer_size) {
        printf("%llu bytes remaining ", (unsigned long long)data->buffer_size);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->buffer_size) * 100) / originalContentLength));
    }
    return toRead;
}
// 创建模拟的流式上传数据
char* generate_upload_buffer(uint64_t buffer_size) {
    void* upload_buffer = NULL;
    if (buffer_size > 0) {
        upload_buffer = malloc(buffer_size);
        if (upload_buffer != NULL) {
            memset(upload_buffer, 't', buffer_size);
        }
    }
    return upload_buffer;
}
```

相关链接

- 关于修改写对象的API说明，请参见**修改写对象**。
- 更多关于修改写对象的代码示例，请参见**Github示例**。
- 修改写对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 修改写对象常见问题请参见**我可以修改对象名称吗？** 。

8 下载对象(C SDK)

8.1 对象下载(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以根据需要将存储在OBS中的对象以文本形式下载到本地。

接口约束

- 您必须是桶拥有者或拥有下载对象的权限，才能下载对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 对于存储类别为归档存储或深度归档存储的对象，需要确认对象的状态为“已恢复”才能对其进行下载。

方法定义

```
void get_object(const obs_options *options, obs_object_info *object_info,
                obs_get_conditions *get_conditions,
                server_side_encryption_params *encryption_params,
                obs_get_object_handler *handler, void *callback_data);
```

请求参数

表 8-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
get_conditions	**obs_get_conditions** *	必选	**参数解释：** 对象筛选条件和读取范围设置。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 获取对象的解密设置。 **约束限制：** 无
handler	**obs_get_object_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 8-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 8-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 8-5 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-6 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 8-7 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 8-8 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 8-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-11 obs_get_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
get_object_data_callback	**obs_get_object_data_callback** *	必选	**参数解释：** 回调函数指针，用于把要下载的数据从数据缓冲区复制到用户自定义回调数据。 **约束限制：** 无

表 8-12 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 8-13 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-14 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 8-15 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-16 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-17 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-18 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 8-19 obs_get_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待下载的数据缓冲区，把这个缓冲区的数据复制到callback_data中，实现下载数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-20 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-21 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 8-22 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-23 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 8-24 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 8-25 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

代码示例：对象下载

以下示例展示如何通过get_object将对象以文件形式下载到本地：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
```

```
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何通过get_object函数下载对象到本地文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 下载起始位置，默认0，从对象0字节位置开始下载
    getcondition.start_byte = 0;
    // 下载长度，默认0，一直下载到对象尾
    // getcondition.byte_count = 0;
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
```



```
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
```



```
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

说明

- 通过操作对象输入流可将对象的内容读取到本地文件或者内存中。
- getcondition可以传空，表示下载整个对象。

8.2 限定条件下载(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

下载对象时，可以指定一个或多个限定条件，满足限定条件时则进行下载，否则返回异常码，下载对象失败。

接口约束

- 您必须是桶拥有者或拥有下载对象的权限，才能下载对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 对于存储类别为归档存储或深度归档存储的对象，需要确认对象的状态为“已恢复”才能对其进行下载。

方法定义

```
void get_object(const obs_options *options, obs_object_info *object_info,
    obs_get_conditions *get_conditions,
    server_side_encryption_params *encryption_params,
    obs_get_object_handler *handler, void *callback_data);
```

请求参数

表 8-26 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
get_conditions	**obs_get_conditions** *	必选	**参数解释：** 对象筛选条件和读取范围设置。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 获取对象的解密设置。 **约束限制：** 无
handler	**obs_get_object_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-27 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 8-28 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 8-29 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 8-30 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-31 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 8-32 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 8-33 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 8-34 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-35 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-36 obs_get_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
get_object_data_callback	obs_get_object_data_callback *	必选	**参数解释：** 回调函数指针，用于把要下载的数据从数据缓冲区复制到用户自定义回调数据。 **约束限制：** 无

表 8-37 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 8-38 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-39 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-40 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-41 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-42 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-43 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 8-44 obs_get_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待下载的数据缓冲区，把这个缓冲区的数据复制到callback_data中，实现下载数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-45 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-46 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 8-47 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-48 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 8-49 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 8-50 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

代码示例：限定条件下载

以下示例展示如何通过get_object限定条件后下载对象到本地：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
```

```
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何通过get_object函数下载对象到本地文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    getcondition.if_match_etag = "34a7ef8cc629583f3cdd882791f31351";
    // 设置下载条件if_modified_since为时间戳0
    getcondition.if_modified_since = 0;
    getcondition.if_not_match_etag = "34a7ef8cc629583f3cdd882791f31350";
    // 设置下载条件if_modified_since为时间戳2147483647
    getcondition.if_not_modified_since = INT_MAX;
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
```



```
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
```



```
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

相关链接

- 关于下载对象的API说明，请参见**获取对象内容**。
- 更多关于下载对象的代码示例，请参见**Github示例**。
- 下载对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 下载对象常见问题请参见**下载对象失败**。

8.3 恢复归档或深度归档存储对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

如果要获取归档或深度归档存储对象的内容，需要先将对象恢复，然后再执行下载数据的操作。对象恢复后，会产生一个标准存储类别的对象副本，也就是说会同时存在标准存储类别的对象副本和归档或深度归档存储对象，在恢复对象的保存时间到期后标准存储类别的对象副本会自动删除。

该接口可以恢复指定桶中的归档或深度归档存储对象。

接口约束

- 您必须是桶拥有者或拥有恢复归档或深度归档存储对象的权限，才能恢复归档或深度归档存储对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:RestoreObject权限，如果使用桶策略则需授予RestoreObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 重复恢复归档或深度归档存储对象时在延长恢复有效期的同时，也将会对恢复时产生的恢复费用进行重复收取。产生的标准存储类别的对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

方法定义

```
void restore_object(const obs_options *options, obs_object_info *object_info, const char *days, obs_tier tier,const obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 8-51 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
days	char *	必选	**参数解释：** 指定恢复对象的保存时间。 **约束限制：** 无 **取值范围：** [1，30]，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
tier	obs_tier	可选	**参数解释：** 恢复选项，可以为：obs_tier.OBS_TIER_EXPEDITED，obs_tier.OBS_TIER_STANDARD。 **约束限制：** 无 **取值范围：** 请详见obs_tier。
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-52 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 8-53 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 8-54 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 8-55 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-56 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 8-57 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 8-58 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 8-59 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-60 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-61 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** ● 同一个桶中存储的对象名必须是唯一的。 ● 指定的对象必须是归档存储类别，否则调用该接口会报错。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-62 obs_tier

枚举值	说明
OBS_TIER_NULL	默认的无效值
OBS_TIER_STANDARD	表示标准恢复对象，归档存储恢复耗时3~5 h，深度归档（受限公测）存储恢复约耗时5~12 h。
OBS_TIER_EXPEDITED	表示快速恢复对象，归档存储恢复耗时1~5 min，深度归档（受限公测）存储恢复约耗时3~5 h。

表 8-63 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 8-64 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-65 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-66 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
server	const char *	可选	**参数解释:** HTTP请求中的Server头域。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-67 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-68 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-69 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：恢复并下载归档存储对象

以下示例展示如何通过restore_object恢复归档对象后，再通过get_object函数下载到本地（请注意恢复时间可能比较长）：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
void wait_until_restore_complete(obs_options* options, obs_object_info* object_info,
server_side_encryption_params* sse);
int main()
{
    // 以下示例展示如何通过get_object函数下载归档存储对象到本地文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
```



```
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test_restore";
    object_info.version_id = NULL;
    // 设置响应回调函数
    obs_response_handler handler =
    {
      &response_properties_callback, &response_complete_callback
    };
    // 恢复对象
    obs_tier tier = OBS_TIER_EXPEDITED;
    obs_status ret_status = OBS_STATUS_BUTT;
    const char* storedDays = "1";
    options.request_options.auth_switch = OBS_OBS_TYPE;
    restore_object(&options, &object_info, storedDays, tier, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("restore object successfully. \n");
    }
    else
    {
        printf("restore object faied(%s).\n", obs_get_status_name(ret_status));
        return;
    }
    // 等待对象恢复完成
    wait_until_restore_complete(&options, &object_info, NULL);
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
```



```
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
typedef struct test_restore_info {
    char restore_info[128];
    obs_status status;
}test_restore_info;
void response_complete_callback_restore(obs_status status, const obs_error_details *error, void
*callback_data)
{
    if (callback_data) {
        test_restore_info *data =
            (test_restore_info *)callback_data;
        data->status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
static obs_status response_properties_callback_restore(const obs_response_properties *properties, void
*callback_data)
{
    if (callback_data) {
        test_restore_info *data =
            (test_restore_info *)callback_data;
        if (properties->restore != NULL) {
            strcpy(data->restore_info, properties->restore);
        }
    }
    else {
        printf("Callback_data is NULL");
    }
}
void wait_until_restore_complete(obs_options* options, obs_object_info* object_info,
server_side_encryption_params* sse) {
    test_restore_info restore_info;
    // 设置响应回调函数
    obs_response_handler handler =
    {
      &response_properties_callback_restore, &response_complete_callback_restore
    };
    do {
        memset(restore_info.restore_info, 0, sizeof(restore_info.restore_info));
        restore_info.status = OBS_STATUS_BUTT;
        get_object_metadata(options, object_info, sse, &handler, &restore_info);
        char* prefix = "ongoing-request=\"false\", expiry-date=";
        // 判断字符串是否以特定字符开头
        if (strstr(restore_info.restore_info, prefix) == restore_info.restore_info) {
            printf("restore_info.restore_info:%s\n", restore_info.restore_info);
            break;
        }
```



```
        else {
            printf("restore_info.restore_info:%s\n", restore_info.restore_info);
        }
        Sleep(6 * 1000);
    } while (1);
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
```



```
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
    print_error_details(error);
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

相关链接

- 关于恢复归档存储对象的API说明，请参见**恢复归档或深度归档存储对象**。
- 更多关于恢复归档存储对象的代码示例，请参见**Github示例**。
- 恢复归档存储对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

8.4 断点续传下载(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

当下载大对象到本地文件时，经常出现因网络不稳定或程序崩溃导致下载失败的情况。失败后再次重新下载不仅浪费资源，而且当网络不稳定时仍然有下载失败的风险。断点续传下载接口能有效地解决此类问题引起的下载失败，其原理是将待下载的对象分成若干个分段分别下载，并实时地将每段下载结果统一记录在checkpoint文件中，仅当所有分段都下载成功时返回下载成功的结果，否则返回错误信息提醒用户再次调用接口进行重新下载（重新下载时因为有checkpoint文件记录当前的下载进度，避免重新下载所有分段，从而节省资源提高效率）。

您可以通过download_file进行断点续传下载。

接口约束

- 您必须是桶拥有者或拥有下载对象的权限，才能下载对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 断点续传下载接口是利用**范围下载**特性实现的，是对范围下载的封装和加强。
- 断点续传下载接口不仅能在失败后重试下载时节省资源提高效率，还因其对分段进行并发下载的机制能加快下载速度，帮助用户快速完成下载业务；且其对用户透明，用户不用关心checkpoint文件的创建和删除、分段任务的切分、并发下载的实现等内部细节。

方法定义

```
void download_file(const obs_options *options, char *key, char* version_id,
    obs_get_conditions *get_conditions,
    server_side_encryption_params *encryption_params,
    obs_download_file_configuration * download_file_config,
    obs_download_file_response_handler *handler, void *callback_data);
```

请求参数说明

表 8-70 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
version_id	char *	必选	**参数解释：** 对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
download_file_config	**obs_download_file_configuration** *	必选	**参数解释：** 下载文件的配置说明。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
get_conditions	obs_get_conditions *	必选	**参数解释：** 对象筛选条件和读取范围设置。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 下载对象解密设置。 **约束限制：** 无
handler	obs_download_file_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-71 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 8-72 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **取值范围：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **取值范围：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **取值范围：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 8-73 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 8-74 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-75 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 8-76 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 8-77 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 8-78 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-79 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-80 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 8-81 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	**image_process_mode_type**	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见**image_process_mode_type**。

参数名称	参数类型	是否必选	描述
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-82 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 8-83 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 8-84 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 8-85 obs_download_file_configuration

参数名称	参数类型	是否必选	描述
downLoad_file	char *	必选	**参数解释：** 要下载到的本地文件路径。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
part_size	uint64_t	可选	**参数解释：** 分段大小，单位字节。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
check_point_file	char *	可选	**参数解释：** 记录上传进度的文件，只在断点续传模式下有效。 **约束限制：** ● 当该值为空时，默认与待上传的本地文件同目录。 ● 该参数仅在enable_check_point参数的值为1时有效。 **取值范围：** 无 **默认取值：** 无
enable_check_point	int	可选	**参数解释：** 是否启用断点续传。 **约束限制：** 无 **取值范围：** ● 0（表示不启用断点续传模式，此时断点续传上传接口退化成对分段上传的简单封装，不会产生checkpoint文件） ● 1（表示启用断点续传模式） **默认取值：** 0（表示不开启）
task_num	int	可选	**参数解释：** 分段上传时的最大并发数。断点续传下载可通过该参数设置并发数，实现对单个对象的并行下载。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 1

表 8-86 obs_download_file_response_handler

参数名称	参数类型	是否必选	描述
response_handler	obs_response_handler *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
download_file_callback	obs_download_file_callback *	必选	**参数解释:** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无

表 8-87 obs_download_file_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释:** 请求状态。 **约束限制:** 无 **取值范围:** 请详见obs_status。
result_message	char *	必选	**参数解释:** 下载结果。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 8-88 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 8-89 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-90 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-91 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000

参数名称	参数类型	是否必选	描述
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-92 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	**obs_name_value***	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
error_headers	char**	**参数解释:** 响应头域中包含error的所有头域。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 8-93 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
value	char *	可选	**参数解释:** 属性的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 8-94 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。

枚举值	说明
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：断点续传下载

以下示例展示如何通过download_file进行断点续传下载对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
void downloadFileResultCallback(obs_status status,
    char *resultMsg,
    int partCountReturn,
    obs_download_file_part_info * downloadInfoList,
    void *callbackData);
int main()
{
    // 以下示例展示如何通过download_file进行断点续传下载：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置下载的对象
    char* key = "example_get_file_test";
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    // 初始化getConditions
    obs_get_conditions getConditions;
    memset(&getConditions, 0, sizeof(obs_get_conditions));
    init_get_properties(&getConditions);
    // 断点下载对象信息
    obs_download_file_configuration downloadFileConfig;
    memset(&downloadFileConfig, 0, sizeof(obs_download_file_configuration));
    // 断点下载对象分段大小
    uint64_t downloadSliceSize = 5L * 1024 * 1024;
    downloadFileConfig.check_point_file = NULL;
    downloadFileConfig.enable_check_point = 1;
    downloadFileConfig.part_size = downloadSliceSize;
    downloadFileConfig.task_num = 10;
    downloadFileConfig.downLoad_file = file_name;
    // 设置响应回调函数
    obs_download_file_response_handler Handler =
    {
        {&response_properties_callback, &response_complete_callback},
        &downloadFileResultCallback
    };
    char* versionId = NULL;
    server_side_encryption_params* sse = NULL;
    initialize_break_point_lock();
    obs_status ret_status = OBS_STATUS_BUTT;
    download_file(&options, key, versionId, &getConditions, sse, &downloadFileConfig,
        &Handler, &ret_status);
    deinitialize_break_point_lock();
    if (OBS_STATUS_OK == ret_status) {
        printf("test download file successfully. \n");
```



```
    }
    else
    {
        printf("test download file faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
```



```
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
//downloadFileResultCallback 示例，printf语句可以替换为自定义的日志打印语句
void downloadFileResultCallback(obs_status status,
    char *resultMsg,
    int partCountReturn,
    obs_download_file_part_info * downloadInfoList,
    void *callbackData)
{
    int i = 0;
    obs_download_file_part_info * pstDownloadInfoList = downloadInfoList;
    printf("status return is %d(%s)\n", status, obs_get_status_name(status));
    printf("%s", resultMsg);
    printf("partCount[%d]\n", partCountReturn);
    for (i = 0; i < partCountReturn; i++)
    {
        printf("partNum[%d],startByte[%llu],partSize[%llu],status[%d]\n",
            pstDownloadInfoList->part_num,
            pstDownloadInfoList->start_byte,
            pstDownloadInfoList->part_size,
            pstDownloadInfoList->status_return);
        pstDownloadInfoList++;
    }
    if (callbackData) {
        obs_status* retStatus = (obs_status*)callbackData;
        (*retStatus) = status;
    }
}
```

相关链接

- 关于下载对象的API说明，请参见**获取对象内容**。

- 更多关于下载对象的代码示例，请参见**Github示例**。
- 下载对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 下载对象常见问题请参见**下载对象失败**。

8.5 图片处理(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS为用户提供了稳定、安全、高效、易用、低成本的图片处理服务。当要下载的对象是图片文件时，您可以通过传入图片处理参数对图片文件进行图片剪切、图片缩放、图片水印、格式转换等处理。

接口约束

- 您必须是桶拥有者或拥有图片处理的权限，才能进行图片处理。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 所有的图片处理操作均不会修改原图。
- 归档存储不支持图片处理。
- 深度归档存储不支持图片处理。
- 处理后的图片直接返回浏览器展示，不会保存在OBS中，也不会占用存储空间，不会产生存储费用。图片处理只收取处理的费用。
- 图片处理参数当前仅支持命令方式，即image/commands格式。
- 图片处理参数支持级联处理，可对图片文件依次实施多条命令。

方法定义

```
void get_object(const obs_options *options, obs_object_info *object_info,
    obs_get_conditions *get_conditions,
    server_side_encryption_params *encryption_params,
    obs_get_object_handler *handler, void *callback_data);
```

请求参数说明

表 8-95 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
get_conditions	**obs_get_conditions** *	必选	**参数解释：** 对象筛选条件和读取范围设置。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 获取对象的解密设置。 **约束限制：** 无
handler	**obs_get_object_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-96 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 8-97 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 8-98 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 8-99 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-100 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 8-101 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 8-102 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 8-103 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 8-104 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-105 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-106 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
image_process_config	image_process_configure *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 8-107 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-108 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 8-109 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 8-110 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 8-111 obs_get_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
get_object_data_callback	**obs_get_object_data_callback** *	必选	**参数解释：** 回调函数指针，用于把要下载的数据从数据缓冲区复制到用户自定义回调数据。 **约束限制：** 无

表 8-112 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 8-113 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-114 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 8-115 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-116 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-117 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 8-118 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 8-119 obs_get_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待下载的数据缓冲区，把这个缓冲区的数据复制到callback_data中，实现下载数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：下载对象接口实现图片处理

以下示例展示如何通过get_object函数实现图片处理并且下载处理后的图片到本地：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何通过get_object函数实现图片处理并且下载处理后对象到本地文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    options.bucket_options.protocol = OBS_PROTOCOL_HTTP;
    // 设置对象下载到本地的文件名
    char *file_name = "./example_image_process_get_file_test.png";
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_image_process_get_file_test.png";
    object_info.version_id = NULL;
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
```



```
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    image_process_configure image_process_config;
    memset(&image_process_config, 0, sizeof(image_process_configure));
    getcondition.image_process_config = &image_process_config;
    getcondition.image_process_config->image_process_mode = obs_image_process_cmd;
    getcondition.image_process_config->cmds_stylename = "resize,m_fixed,w_100,h_100/rotate,90";
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
```



```
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

相关链接

- 关于图片处理的更多信息，请参见**图片处理**。
- 更多关于图片处理的代码示例，请参见**Github示例**。
- 图片处理过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 图片处理常见问题请参见**图片处理常见问题**。

9 管理对象(C SDK)

9.1 设置对象属性(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以在上传对象时设置对象属性。对象属性包含对象长度、对象MIME类型、对象MD5值（用于校验）、对象存储类别、对象自定义元数据。对象属性可以在多种上传方式下（流式上传、文件上传、分段上传），或**复制对象**时进行设置。

接口约束

- 您必须是桶拥有者或拥有设置对象属性的权限，才能设置对象属性。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:ModifyObjectMetaData权限，如果使用桶策略则需授予ModifyObjectMetaData权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个对象可以有多个元数据，总大小不能超过8KB。
- 当前元数据名称不支持非ASCII码字符，元数据值包含非ASCII码字符时需进行base64编码。
- 如果不设置存储类别，对象的存储类别默认与桶的存储类别保持一致。

方法定义

```
void set_object_metadata(const obs_options *options, obs_object_info *object_info,
    obs_put_properties *put_properties,
    server_side_encryption_params *encryption_params,
    obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 9-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
put_properties	**obs_put_properties***	必选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 请详见**创建访问密钥**。 **取值范围：** 无 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 9-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-5 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-6 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-7 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-8 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-11 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-12 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
get_conditions	obs_get_conditions *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	**obs_name_value***	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	**metadata_action_indicator**	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见**metadata_action_indicator**。

参数名称	参数类型	是否必选	描述
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 9-13 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 9-14 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 9-15 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 9-16 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-17 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	**image_process_mode_type**	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见**image_process_mode_type**。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-18 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。

枚举值	说明
obs_image_process_style	图片处理参数以style开头。

表 9-19 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围：** ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 9-20 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 9-21 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-22 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-23 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 9-24 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-25 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-26 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-27 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置对象 MIME 类型

以下示例展示在文件上传对象时如何设置对象MIME类型（如果不设置对象MIME类型，C SDK会根据上传对象的后缀名自动判断对象MIME类型，如.xml判断为application/xml文件；.html判断为text/html文件。）：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下代码以文件上传为例展示如何设置对象MIME类型
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
```



```
SECRET_ACCESS_KEY
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_test";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    // 设置MIME类型
    put_properties.content_type = "text/plain";
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
```



```
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
```



```
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

代码示例：设置对象存储类别

以下示例展示如何在上传对象时设置对象存储类别：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何在上传对象时设置对象存储类别：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    //设置存储类别为归档存储
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_GLACIER;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_test_cold";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
```



```
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
```



```
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

代码示例：设置对象自定义元数据

以下示例展示如何在上传对象时设置对象自定义元数据：


```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何在上传对象时设置对象自定义元数据：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_test_with_matadatas";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    //设置自定义元数据
    obs_name_value example_matadatas[2] = {0};
    example_matadatas[0].name = "example-property1";
    example_matadatas[0].value = "example-property-value1";
    example_matadatas[1].name = "example-property2";
    example_matadatas[1].value = "example-property-value2";
    put_properties.meta_data = &example_matadatas;
    //设置自定义元数据数量
    put_properties.meta_data_count = 2;
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
```



```
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
```



```
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

相关链接

- 关于上传对象-PUT上传的API说明，请参见**PUT上传**。
- 更多关于上传对象的代码示例，请参见**Github示例**。
- 上传对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 上传对象常见问题请参见**上传对象失败**。

9.2 获取对象属性(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

元数据（Metadata）为描述对象属性的信息，是一组名称值对，用作对象管理的一部分。

当前仅支持系统定义的元数据。系统定义的元数据又分为两种类别：系统控制和用户控制。如Last-Modified日期等数据由系统控制，不可修改；如为对象配置的ContentLanguage，用户可以通过接口进行修改。

对指定桶中的对象发送HEAD请求，获取对象的元数据信息。

接口约束

- 您必须是桶拥有者或拥有获取对象元数据的权限，才能获取对象元数据。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_object_metadata(const obs_options *options, obs_object_info *object_info,
    server_side_encryption_params *encryption_params,
    obs_response_handler *handler, void *callback_data);
```

请求参数

表 9-28 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
object_info	obs_object_info *	必选	**参数解释：** 对象名和版本号，非多版本对象，version设置为0。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-29 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-30 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-31 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-32 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-33 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-34 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-35 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-36 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-37 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-38 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-39 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 9-40 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 9-41 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-42 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties** *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-43 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-44 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** 请求中的Server头域 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-45 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-46 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-47 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取对象属性

以下示例展示如何通过get_object_metadata来获取对象元数据：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过get_object_metadata来获取对象元数据：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 对象信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    // 设置响应回调函数
```



```
    obs_response_handler response_handler =
    {
        // 对象属性可以在response_properties_callback中获取
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 获取对象属性
    get_object_metadata(&options, &object_info, 0, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("get object metadata successfully. \n");
    }
    else
    {
        printf("get object metadata failed.\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
```



```
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

9.3 设置对象 ACL-上传对象时指定预定义访问策略(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS支持对对象的操作进行权限控制。默认情况下，只有对象的创建者才有该对象的读写权限。用户也可以设置其他的访问策略，比如对一个对象可以设置公共访问策

略，允许所有人对其都有读权限。SSE-KMS方式加密的对象即使设置了ACL，跨租户也不生效。

OBS用户在上传对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的对象更改或者获取ACL（access control list）。

可以通过本接口在上传对象时指定预定义访问策略。

接口约束

- 您必须是桶拥有者或拥有设置对象ACL的权限，才能设置对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObjectAcl权限，如果使用桶策略则需授予PutObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个对象的ACL最多支持配置100条授权策略。

方法定义

```
void put_object(const obs_options *options, char *key, uint64_t content_length,
                obs_put_properties *put_properties,
                server_side_encryption_params *encryption_params,
                obs_put_object_handler *handler, void *callback_data);
```

请求参数说明

表 9-48 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。
put_properties	**obs_put_properties***	必选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_put_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-49 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-50 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-51 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-52 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-53 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-54 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-55 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-56 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-57 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-58 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（指从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。 **默认取值：** 无
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD %E6%96%87: %E4%B8%AD %E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。 **默认取值：** 无
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 9-59 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** **0（指从对象的第一个字节开始下载）**
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** 无 **取值范围：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 9-60 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 9-61 obs_az_redundancy

枚举值	说明
OBS_REDUNDANCY_1AZ	默认创建单AZ桶。
OBS_REDUNDANCY_3AZ	创建3AZ桶。

表 9-62 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。

枚举值	说明
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 9-63 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。 **约束限制：** 无 **取值范围：** • application/x-www-form-urlencoded • application/json **默认取值：** 如果不设置取值，则默认为application/json。

表 9-64 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-65 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 9-66 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见查看密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 9-67 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。

枚举值	说明
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 9-68 obs_put_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释:** 响应回调函数结构体。 **约束限制:** 无
put_object_data_callback	**obs_put_object_data_callback** *	必选	**参数解释:** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制:** 无
progress_callback	**obs_progress_callback_internal** *	必选	**参数解释:** 进度回调函数指针。 **约束限制:** 无

表 9-69 obs_put_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释:** buffer的长度。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
buffer	char *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-70 obs_progress_callback_internal

参数名称	参数类型	是否必选	描述
now	uint64_t	必选	**参数解释：** 已上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
total	uint64_t	必选	**参数解释：** 总共需要上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-71 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-72 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-73 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-74 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。如果该对象无版本号，则为NULL。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-75 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-76 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-77 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：上传对象时指定预定义访问策略

以下示例展示如何在上传对象时指定预定义访问策略

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void put_buffer_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_buffer_object_callback_data
{
    char *put_buffer;
    uint64_t buffer_size;
    uint64_t cur_offset;
    obs_status ret_status;
} put_buffer_object_callback_data;
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data);
char* generate_upload_buffer(uint64_t buffer_size);
int main()
{
    // 以下示例展示如何在上传对象时指定预定义访问策略
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
```



```
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 上传的对象名
    char *key = "example_put_buffer_test_with_acl.txt";
    // 初始化结构体put_properties，可以通过该结构体设置对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 指定预定义访问策略
    put_properties.canned_acl = OBS_CANNED_ACL_PUBLIC_READ_WRITE;
    // 回调数据
    put_buffer_object_callback_data data;
    memset(&data, 0, sizeof(put_buffer_object_callback_data));
    // 设置buffersize
    data.buffer_size = 10 * 1024 * 1024;
    // 创建模拟的流式上传数据buffer, 并赋值到上传数据结构中
    data.put_buffer = generate_upload_buffer(data.buffer_size);
    // 回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_buffer_complete_callback },
        &put_buffer_data_callback
    };
    // 上传数据流
    put_object(&options, key, data.buffer_size, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
```



```
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void put_buffer_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        put_buffer_object_callback_data *data = (put_buffer_object_callback_data *)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
```



```
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
int put_buffer_data_callback(int buffer_size, char *buffer, void *callback_data)
{
    put_buffer_object_callback_data *data =
        (put_buffer_object_callback_data *)callback_data;
    int toRead = 0;
    if (data->buffer_size) {
        toRead = ((data->buffer_size > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->buffer_size);
        memcpy_s(buffer, buffer_size, data->put_buffer + data->cur_offset, toRead);
    }
    uint64_t originalContentLength = data->buffer_size;
    data->buffer_size -= toRead;
    data->cur_offset += toRead;
    if (data->buffer_size) {
        printf("%llu bytes remaining ", (unsigned long long)data->buffer_size);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->buffer_size) * 100) / originalContentLength));
    }
    return toRead;
}
// 创建模拟的流式上传数据
char* generate_upload_buffer(uint64_t buffer_size) {
    void* upload_buffer = NULL;
    if (buffer_size > 0) {
        upload_buffer = malloc(buffer_size);
        if (upload_buffer != NULL) {
            memset(upload_buffer, 't', buffer_size);
        }
    }
    return upload_buffer;
}
```

相关链接

- 关于设置对象ACL的API说明，请参见**设置对象ACL**。
- 更多关于设置对象ACL的示例代码，请参见**Github示例**。
- 设置对象ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.4 设置对象 ACL-为对象设置预定义访问策略(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS支持对对象的操作进行权限控制。默认情况下，只有对象的创建者才有该对象的读写权限。用户也可以设置其他的访问策略，比如对一个对象可以设置公共访问策略，允许所有人对其都有读权限。SSE-KMS方式加密的对象即使设置了ACL，跨租户也不生效。

OBS用户在上传对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的对象更改或者获取ACL（access control list）。

可以通过本接口为指定桶中对象设置预定义访问策略。

接口约束

- 您必须是桶拥有者或拥有设置对象ACL的权限，才能设置对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObjectAcl权限，如果使用桶策略则需授予PutObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个对象的ACL最多支持配置100条授权策略。

方法定义

```
void set_object_acl_by_head(const obs_options *options, obs_object_info *object_info,
    obs_canned_acl canned_acl, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 9-78 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
canned_acl	**obs_canned_acl**	必选	**参数解释：** 预定义访问策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-79 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-80 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-81 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-82 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-83 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-84 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-85 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-86 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-87 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-88 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-89 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-90 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-91 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-92 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-93 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-94 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-95 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-96 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

代码示例：为对象设置预定义访问策略

以下示例展示如何通过set_object_acl_by_head来设置对象属性：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过set_object_acl_by_head来设置对象属性：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
```



```
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    obs_canned_acl canned_acl = OBS_CANNED_ACL_PUBLIC_READ_WRITE;
    obs_object_info object_info;
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    // 设置对象预定义访问策略
    set_object_acl_by_head(&options, &object_info, canned_acl, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set object acl by head successfully. \n");
    }
    else
    {
        printf("set object acl by head failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
```



```
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

相关链接

- 关于设置对象ACL的API说明，请参见**设置对象ACL**。
- 更多关于设置对象ACL的示例代码，请参见**Github示例**。
- 设置对象ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.5 设置对象 ACL-直接设置对象 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS支持对对象的操作进行权限控制。默认情况下，只有对象的创建者才有该对象的读写权限。用户也可以设置其他的访问策略，比如对一个对象可以设置公共访问策略，允许所有人对其都有读权限。SSE-KMS方式加密的对象即使设置了ACL，跨租户也不生效。

OBS用户在上传对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的对象更改或者获取ACL（access control list）。

可以通过本接口为指定桶中对象直接设置对象的访问权限。

接口约束

- 您必须是桶拥有者或拥有设置对象ACL的权限，才能设置对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObjectAcl权限，如果使用桶策略则需授予PutObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个对象的ACL最多支持配置100条授权策略。

方法定义

```
void set_object_acl(const obs_options *options, manager_acl_info *aclinfo, obs_response_handler *handler,
void *callback_data);
```

请求参数说明

表 9-97 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无

参数名称	参数类型	是否必选	描述
aclinfo	manager_acl_info *	必选	**参数解释：** 管理ACL权限信息相关结构体。 **约束限制：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-98 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-99 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-100 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-101 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-102 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-103 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-104 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-105 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-106 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-107 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-108 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-109 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-110 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-111 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-112 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-113 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-114 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 如果非多版本对象，请把version设置为NULL
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。 **默认取值：** OBJECT_DELIVERED_TRUE（指对象ACL继承桶的ACL）
acl_grants	**obs_acl_grant** *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 9-115 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-116 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 9-117 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	obs_grantee_type	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见obs_grantee_type。

参数名称	参数类型	是否必选	描述
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 9-118 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 9-119 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 9-120 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

代码示例：直接设置对象访问权限

以下示例展示如何通过set_object_acl来设置对象访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过set_object_acl来设置对象访问权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
```



```
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 定义对象访问权限信息
    manager_acl_info aclinfo;
    memset(&aclinfo, 0, sizeof(aclinfo));
    aclinfo.object_info.key = "example_get_file_test";
    aclinfo.object_info.version_id = NULL;
    aclinfo.owner_id = "1****************************a";
    // 设置acl
    int acl_grant_count = 1;
    aclinfo.acl_grant_count_return = &acl_grant_count;
    aclinfo.acl_grants = (obs_acl_grant*)malloc(sizeof(obs_acl_grant) * acl_grant_count);
    // 设置被授权账号的租户ID(canonical_user.id)
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.id, "0****************************0");
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.display_name, "name1");
    aclinfo.acl_grants[0].grantee_type = OBS_GRANTEE_TYPE_CANONICAL_USER;
    // 设置ACL权限，此处以读权限为例
    aclinfo.acl_grants[0].permission = OBS_PERMISSION_READ;
    // 设置对象访问权限
    set_object_acl(&options, &aclinfo, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set object acl successfully. \n");
    }
    else
    {
        printf("set object acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    free(aclinfo.acl_grants);
    aclinfo.acl_grants = NULL;
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
```

```
    do {                                                \
        if (properties-> field) {                       \
            printf("%s: %s\n", name, properties->field);        \
        }                                               \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
```



```
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

相关链接

- 关于设置对象ACL的API说明，请参见**设置对象ACL**。
- 更多关于设置对象ACL的示例代码，请参见**Github示例**。
- 设置对象ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.6 获取对象 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持对对象操作进行权限控制，您可以为对象设置访问策略，指定某一个用户对某一个对象是否有权行使某一项指定操作。OBS权限控制的方式有IAM、桶策略和ACL三种，ACL按照粒度又分为桶ACL和对象ACL，本节将对对象ACL接口进行详细介绍，更多权限相关内容可参见《对象存储服务权限配置指南》的**OBS权限控制概述**章节。

对象ACL是跨账号场景的权限，设置授权的对象不是当前账号，也不是当前账号下的IAM用户，而是另一个华为云账号及其账号下的IAM用户；授权的范围是以对象为粒度的，一条ACL策略为一个对象设置策略，因此设置ACL策略时您必须明确指定对象名；对象ACL授予的权限包括对象的访问权限和对象ACL的访问权限两个方面，对象的访问权限包括对桶内对象的查看和编辑权限，桶ACL的访问权限包括对桶ACL策略的查看和编辑权限，详情可参见**ACL权限控制方式介绍**。

调用获取对象ACL接口，您可以获取指定对象的ACL策略。

接口约束

- 您必须是桶拥有者或拥有获取对象ACL的权限，才能获取对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObjectAcl权限，如果使用桶策略则需授予GetObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的region以及region与endPoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_object_acl(const obs_options *options, manager_acl_info *aclinfo, obs_response_handler *handler,
void *callback_data);
```

请求参数说明

表 9-121 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无
aclinfo	**manager_acl_info** *	必选	**参数解释：** 管理ACL权限信息相关结构体。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-122 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-123 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 9-124 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-125 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-126 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-127 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-128 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-129 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 9-130 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-131 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-132 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-133 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-134 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-135 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-136 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-137 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-138 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 如果非多版本对象，请把version设置为NULL
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。 **默认取值：** OBJECT_DELIVERED_TRUE（指对象ACL继承桶的ACL）
acl_grants	**obs_acl_grant** *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 9-139 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-140 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 9-141 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	obs_grantee_type	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见obs_grantee_type。

参数名称	参数类型	是否必选	描述
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 9-142 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 9-143 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 9-144 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

代码示例：获取对象访问权限

以下示例展示如何通过get_object_acl获取对象的访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
manager_acl_info* malloc_acl_info();
void free_acl_info(manager_acl_info **acl);
void print_acl(manager_acl_info* acl);
int main()
{
    // 以下示例展示如何通过get_object_acl获取对象的访问权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
```



```
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    options.bucket_options.protocol = OBS_PROTOCOL_HTTP;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 对象访问权限信息
    manager_acl_info *aclinfo = malloc_acl_info();
    aclinfo->object_info.key = "example_get_file_test";
    aclinfo->object_info.version_id = NULL;
    // 获取对象权限信息
    get_object_acl(&options, aclinfo, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("get object acl successfully. \n");
        print_acl(aclinfo);
    }
    else
    {
        printf("get object acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 销毁内存
    free_acl_info(&aclinfo);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
```



```
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
```

```
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
void print_acl(manager_acl_info* acl) {
    if (acl == NULL) {
        printf("acl is NULL.\n");
        return;
    }
    printf("object key :%s\n", acl->object_info.key);
    printf("object version_id :%s\n", acl->object_info.version_id);
    printf("object owner_id :%s\n", acl->owner_id);
    printf("object owner_display_name :%s\n", acl->owner_display_name);
    switch (acl->object_delivered)
    {
    case OBJECT_DELIVERED_TRUE:
        printf("object acl delivered.\n");
        break;
    default:
        printf("object acl not delivered.\n");
        break;
    }
    if (acl->acl_grant_count_return == NULL) {
        printf("acl_grant_count_return is NULL.\n");
        return;
    }
    printf("object acl_grant_count :%d\n", *acl->acl_grant_count_return);
    if (acl->acl_grants == NULL) {
        printf("acl_grants is NULL.\n");
        return;
    }
    for (int i = 0; i < *acl->acl_grant_count_return; ++i) {
        obs_acl_grant* acl_grant = acl->acl_grants + i;
        printf("acl %d\n", i + 1);
        switch (acl_grant->bucket_delivered)
        {
        case BUCKET_DELIVERED_TRUE:
            printf("object acl delivered from bucket.\n");
            break;
        default:
            printf("object acl not delivered from bucket.\n");
            break;
        }
        switch (acl_grant->grantee_type)
        {
        case OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL.\n");
            break;
        case OBS_GRANTEE_TYPE_CANONICAL_USER:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_CANONICAL_USER.\n");
            break;
        case OBS_GRANTEE_TYPE_ALL_OBS_USERS:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_ALL_OBS_USERS.\n");
            break;
        case OBS_GRANTEE_TYPE_ALL_USERS:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_ALL_USERS.\n");
            break;
        case OBS_GRANTEE_TYPE_LOG_DELIVERY:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_LOG_DELIVERY.\n");
            break;
        default:
            printf("object acl grantee_type is unknown.\n");
            break;
        }
        switch (acl_grant->permission)
        {
        case OBS_PERMISSION_READ:
            printf("object acl grantee_type is OBS_PERMISSION_READ.\n");
            break;
```



```
        case OBS_PERMISSION_WRITE:
            printf("object acl grantee_type is OBS_PERMISSION_WRITE.\n");
            break;
        case OBS_PERMISSION_READ_ACP:
            printf("object acl grantee_type is OBS_PERMISSION_READ_ACP.\n");
            break;
        case OBS_PERMISSION_WRITE_ACP:
            printf("object acl grantee_type is OBS_PERMISSION_WRITE_ACP.\n");
            break;
        case OBS_PERMISSION_FULL_CONTROL:
            printf("object acl grantee_type is OBS_PERMISSION_FULL_CONTROL.\n");
            break;
        default:
            printf("object acl grantee_type is unknown.\n");
            break;
        }
        printf("acl_grant->grantee.canonical_user.display_name: %s\n", acl_grant-
>grantee.canonical_user.display_name);
        printf("acl_grant->grantee.canonical_user.id: %s\n", acl_grant->grantee.canonical_user.id);
    }
}
manager_acl_info* malloc_acl_info()
{
    manager_acl_info *aclinfo = (manager_acl_info*)malloc(sizeof(manager_acl_info));
    memset(aclinfo, 0, sizeof(manager_acl_info));
    size_t acl_grants_size = OBS_MAX_ACL_GRANT_COUNT * sizeof(obs_acl_grant);
    aclinfo->acl_grants = (obs_acl_grant*)malloc(acl_grants_size);
    memset(aclinfo->acl_grants, 0, acl_grants_size);
    aclinfo->acl_grant_count_return = (int*)malloc(sizeof(int));
    *(aclinfo->acl_grant_count_return) = 0;
    size_t owner_id_size = (OBS_MAX_GRANTEE_USER_ID_SIZE + 1) * sizeof(char);
    size_t owner_display_name_size = (OBS_MAX_GRANTEE_USER_ID_SIZE + 1) * sizeof(char);
    aclinfo->owner_id = (char *)malloc(owner_id_size);
    memset(aclinfo->owner_id, 0, owner_id_size);
    aclinfo->owner_display_name = (char *)malloc(owner_display_name_size);
    memset(aclinfo->owner_display_name, 0, owner_display_name_size);
    return aclinfo;
}
void free_acl_info(manager_acl_info **acl)
{
    manager_acl_info *aclinfo = *acl;
    free(aclinfo->acl_grants);
    free(aclinfo->owner_display_name);
    free(aclinfo->owner_id);
    free(aclinfo->acl_grant_count_return);
    free(aclinfo);
}
```

相关链接

- 关于获取对象ACL的API说明，请参见**获取对象ACL**。
- 更多关于设置对象ACL的示例代码，请参见**Github示例**。
- 设置对象ACL过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.7 列举桶内对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

调用列举桶内对象接口，可列举指定桶内的部分或所有对象的描述信息。您还可以通过设置前缀、数量、起始位置等参数，返回符合您筛选条件的对象信息。返回结果以对象名的字典序排序。

接口约束

- 每次接口调用最多返回1000个对象信息。
- 您必须是桶拥有者或拥有列举桶内对象的权限，才能列举桶内对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:ListBucket权限，如果使用桶策略则需授予ListBucket权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void list_bucket_objects(const obs_options *options, const char *prefix, const char *marker,
    const char *delimiter, int maxkeys, obs_list_objects_handler *handler, void *callback_data);
```

请求参数说明

表 9-145 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
prefix	char *	可选	**参数解释：** 限定返回的对象名必须带有prefix前缀。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
marker	char *	可选	**参数解释：** 列举对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
delimiter	char *	可选	**参数解释：** 用于对对象名进行分组的字符。对于对象名中包含delimiter的对象，其对象名（如果请求中指定了prefix，则此处的对象名需要去掉prefix）中从首字符至第一个delimiter之间的字符串将作为一个分组并作为commonPrefix返回。 对于并行文件系统，不携带此参数时默认列举是递归列举此目录下所有内容，会列举子目录。在大数据场景下（目录层级深、目录下文件多）的列举，建议设置[delimiter="/"]，只列举当前目录下的内容，不列举子目录，提高列举效率。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
maxkeys	int	必选	**参数解释：** 列举对象的最大数目。 **约束限制：** 当该参数超出1000时，按照默认的1000进行处理。 **取值范围：** [1，1000]，单位：个。 **默认取值：** 1000

参数名称	参数类型	是否必选	描述
handler	obs_list_objects_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-146 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-147 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-148 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-149 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-150 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-151 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-152 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-153 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-154 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-155 obs_list_objects_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
list_Objects_callback	obs_list_objects_callback *	必选	**参数解释：** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制：** 无

表 9-156 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-157 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-158 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-159 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-160 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-161 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-162 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-163 obs_list_objects_callback

参数名称	参数类型	是否必选	描述
is_truncated	int	必选	**参数解释：** 结果是否截断。 **约束限制：** 无 **取值范围：** ● 0（代表已返回了全部结果） ● 1（代表没有返回全部结果） **默认取值：** 无

参数名称	参数类型	是否必选	描述
next_marker	const char *	必选	**参数解释：** 下次列举对象请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的key_marker值。 **约束限制：** 该字段仅用于多版本列举。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
contents_count	const char *	必选	**参数解释**： 下次列举象请求的起始位置，与next_key_marker配合使用。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的version_id_marker值。 **约束限制：** 该字段只适用于多版本列举场景。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
contents	const **obs_list_objects_content** *	必选	**参数解释：** 列举出的对象相关信息。 **约束限制：** 无
common_prefixes_count	int	必选	**参数解释：** common_prefixes的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
common_prefixes	const char **	必选	**参数解释：** 当请求中设置了delimiter分组字符时，返回按delimiter分组后的对象名称前缀列表。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-164 obs_list_objects_content

参数名称	参数类型	是否必选	描述
key	const char *	可选	**参数解释：** 对象名。对象名是对象在存储桶中的唯一标识。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
last_modified	int64_t	可选	**参数解释：** 对象最近一次被修改的时间（UTC时间）。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
etag	const char *	可选	**参数解释**： 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。实际的ETag是对象的哈希值。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，通过MD5加密后都有唯一的ETag。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
size	uint64_t	可选	**参数解释：** 对象的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
owner_id	const char *	可选	**参数解释：** 对象拥有者的domain_id（账号ID）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
type	const char *	可选	**参数解释：** 对象类型，非Normal对象时返回。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：列举对象

以下示例展示如何通过函数list_bucket_objects列举出桶里的对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void listobjects_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct list_object_callback_data
{
    int is_truncated;
    char next_marker[1024];
    int keyCount;
    int allDetails;
    obs_status ret_status;
} list_object_callback_data;
obs_status list_objects_callback(int is_truncated, const char *next_marker,
```



```
    int contents_count,
    const obs_list_objects_content *contents,
    int common_prefixes_count,
    const char **common_prefixes,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过函数list_bucket_objects列举出桶里的对象。
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置响应回调函数
    obs_list_objects_handler list_bucket_objects_handler =
    {
        { &response_properties_callback, &listobjects_complete_callback },
        &list_objects_callback
    };
    // 用户自定义回调数据
    list_object_callback_data data;
    memset(&data, 0, sizeof(list_object_callback_data));
    data.allDetails = 1;
    const char* prefix = NULL;
    const char* marker = NULL;
    const char* delimiter = "/";
    int maxkeys = 1000;
    // 列举对象
    list_bucket_objects(&options, prefix, marker, delimiter, maxkeys, &list_bucket_objects_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("list bucket objects successfully. \n");
    }
    else
    {
        printf("list bucket objects failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
```



```
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void listobjects_complete_callback(obs_status status,
```



```
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data)
    {
        list_object_callback_data *data = (list_object_callback_data *)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
static void printListBucketHeader(int allDetails)
{
    printf("%-50s  %-20s  %-5s",
        "   Key",
        "   Last Modified", "Size");
    if (allDetails) {
        printf("  %-34s  %-64s  %-12s           %-12s",
            "   ETag",
            "   Owner ID",
            "Display Name",
            "StorageClass");
    }
    printf("\n");
    printf("--------------------------------------------------  "
        "--------------------  -----");
    if (allDetails) {
        printf("  ----------------------------------  "
            "----------------------------------------------------"
            "---------------  ------------           ------------");
    }
    printf("\n");
}
obs_status list_objects_callback(int is_truncated, const char *next_marker,
    int contents_count,
    const obs_list_objects_content *contents,
    int common_prefixes_count,
    const char **common_prefixes,
    void *callback_data)
{
    list_object_callback_data *data = (list_object_callback_data *)callback_data;
    data->is_truncated = is_truncated;
    if ((!next_marker || !next_marker[0]) && contents_count) {
        next_marker = contents[contents_count - 1].key;
    }
    if (next_marker) {
        snprintf(data->next_marker, sizeof(data->next_marker), "%s",
            next_marker);
    }
    else {
        data->next_marker[0] = 0;
    }
    if (contents_count && !data->keyCount) {
        printListBucketHeader(data->allDetails);
    }
    int i;
    for (i = 0; i < contents_count; i++) {
        const obs_list_objects_content *content = &(contents[i]);
        char timebuf[256] = { 0 };
        time_t t = (time_t)content->last_modified;
        strftime(timebuf, sizeof(timebuf), "%Y-%m-%dT%H:%M:%SZ",
            gmtime(&t));
        char sizebuf[16] = { 0 };
        if (content->size < 100000) {
            sprintf_s(sizebuf, sizeof(sizebuf), "%5llu", (unsigned long long) content->size);
        }
        else if (content->size < (1024 * 1024)) {
```



```
            sprintf_s(sizebuf, sizeof(sizebuf), "%4lluK",
                ((unsigned long long) content->size) / 1024ULL);
        }
        else if (content->size < (10 * 1024 * 1024)) {
            float f = content->size;
            f /= (1024 * 1024);
            sprintf_s(sizebuf, sizeof(sizebuf), "%1.2fM", f);
        }
        else if (content->size < (1024 * 1024 * 1024)) {
            sprintf_s(sizebuf, sizeof(sizebuf), "%4lluM",
                ((unsigned long long) content->size) /
                (1024ULL * 1024ULL));
        }
        else {
            float f = (content->size / 1024);
            f /= (1024 * 1024);
            sprintf_s(sizebuf, sizeof(sizebuf), "%1.2fG", f);
        }
        printf("%-50s  %s  %s", content->key, timebuf, sizebuf);
        if (data->allDetails) {
            printf("  %-36s  %-64s  %-20s  %-16s  %-16s",
                content->etag,
                content->owner_id ? content->owner_id : "",
                content->owner_display_name ? content->owner_display_name : "",
                content->storage_class ? content->storage_class : "",
                content->type ? content->type : ""
            );
        }
        printf("\n");
    }
    data->keyCount += contents_count;
    for (i = 0; i < common_prefixes_count; i++) {
        printf("\nCommon Prefix: %s\n", common_prefixes[i]);
    }
    printf("contents_count:%d\n", contents_count);
    return OBS_STATUS_OK;
}
```

相关链接

- 关于列举桶内对象的API说明，请参见**列举桶内对象**。
- 更多关于列举对象的代码示例，请参见**Github示例**。
- 列举桶内对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

9.8 删除对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

为节省空间和成本，您可以根据需要删除指定桶中的单个对象。

接口约束

- 您必须是桶拥有者或拥有删除对象的权限，才能删除对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:DeleteObject权限，如果使用桶策略则需授予DeleteObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 桶没有开启多版本控制功能时，已删除的对象不可恢复，请谨慎操作。

方法定义

```
void delete_object(const obs_options *options, obs_object_info *object_info,
    obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 9-165 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-166 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure** *	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-167 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-168 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-169 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-170 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-171 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-172 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-173 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-174 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-175 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-176 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-177 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-178 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-179 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-180 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-181 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-182 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释:** 对象名。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
version_id	char *	可选	**参数解释:** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

代码示例：删除对象

以下示例展示如何通过delete_object删除单个对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过delete_object删除单个对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 要删除的对象信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test_delete";
    object_info.version_id = NULL;
    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 删除对象
    delete_object(&options, &object_info, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("delete object successfully. \n");
    }
    else
    {
        printf("delete object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
```



```
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
```



```
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

相关链接

- 关于删除对象的API说明，请参见**删除对象**。
- 更多关于删除对象的代码示例，请参见**Github示例**。
- 删除对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.9 批量删除对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

为节省空间和成本，您可以根据需要删除指定桶中的多个对象。

批量删除对象特性用于将一个桶内的部分对象一次性删除，删除后不可恢复。批量删除对象要求返回结果里包含每个对象的删除结果。OBS的批量删除对象使用同步删除对象的方式，每个对象的删除结果返回给请求用户。

接口约束

- 您必须是桶拥有者或拥有批量删除对象的权限，才能批量删除对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:DeleteObject权限，如果使用桶策略则需授予DeleteObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 桶没有开启多版本控制功能时，已删除的对象不可恢复，请谨慎操作。
- 批量删除对象一次能接收最大对象数目为1000个，如果超出限制，服务端会返回请求不合法。
- 并发任务分配后，在循环删除多个对象过程中， 如果发生内部错误， 有可能出现数据不一致的情况（ 某个对象索引数据删除但还有元数据 ）。

方法定义

```
void batch_delete_objects(const obs_options *options, obs_object_info *object_info,
    obs_delete_object_info *delobj,
    obs_put_properties *put_properties,
    obs_delete_object_handler *handler,
    void *callback_data);
```

请求参数说明

表 9-183 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
delobj	**obs_delete_object_info***	必选	**参数解释：** 删除对象个数和指定使用quiet模式。 **约束限制：** 无
put_properties	**obs_put_properties***	可选	**参数解释：** 设置删除对象的校验属性。 **约束限制：** 无
handler	**obs_delete_object_handler***	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-184 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-185 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-186 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-187 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-188 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-189 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-190 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-191 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-192 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-193 obs_delete_object_info

参数名称	参数类型	是否必选	描述
keys_number	unsigned int	必选	**参数解释：** 删除对象名个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
quiet	int	可选	**参数解释：** 指定是否使用quiet模式。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-194 obs_delete_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
delete_object_data_callback	**obs_delete_object_data_callback** *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-195 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-196 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-197 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-198 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-199 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-200 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-201 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-202 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-203 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
get_conditions	obs_get_conditions *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** • 取值必须大于0。 • 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	**obs_name_value***	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	**metadata_action_indicator**	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见**metadata_action_indicator**。

参数名称	参数类型	是否必选	描述
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 9-204 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 9-205 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 9-206 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 9-207 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-208 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	**image_process_mode_type**	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见**image_process_mode_type**。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-209 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。

枚举值	说明
obs_image_process_style	图片处理参数以style开头。

表 9-210 obs_delete_object_data_callback

参数名称	参数类型	是否必选	描述
contents_count	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
contents	obs_delete_objects *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-211 obs_delete_objects

参数名称	参数类型	是否必选	描述
key	constchar*	必选	**参数解释：** 每个删除结果的对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
code	constchar*	可选	**参数解释：** 删除失败结果的错误码。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
message	constchar*	可选	**参数解释：** 删除失败结果的错误消息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
delete_marker	constchar*	可选	**参数解释：** 当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，返回消息中该元素的值为true。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
delete_marker_version_id	constchar*	可选	**参数解释：** 请求创建或删除的删除标记版本号。 当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，响应消息会返回该元素。该元素在以下两种情况中会出现： • 用户发送不带版本删除请求，即请求只有对象名，无版本号。这种情况下，系统会创建一个删除标记，并在响应中返回该删除标记的版本号。 • 用户发送带版本删除请求，即请求同时包含对象名以及版本号，但是该版本号标识一个删除标记。这种情况下，系统会删除此删除标记，并在响应中返回该删除标记的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：批量删除

以下示例展示如何通过batch_delete_objects删除多个对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
obs_status delete_objects_data_callback(int contentsCount,
```



```
    obs_delete_objects *delobjs,
    void *callbackData);
int main()
{
    // 以下示例展示如何通过batch_delete_objects删除多个对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 要批量删除的对象信息
    obs_object_info objectinfo[100];
    objectinfo[0].key = "example_get_file_test_delete1";
    objectinfo[0].version_id = NULL;
    objectinfo[1].key = "example_get_file_test_delete2";
    objectinfo[1].version_id = "versionid2";
    obs_delete_object_info delobj;
    memset(&delobj, 0, sizeof(obs_delete_object_info));
    delobj.keys_number = 2;
    // 设置响应回调函数
    obs_delete_object_handler handler =
    {
        {&response_properties_callback, &response_complete_callback},
        &delete_objects_data_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 批量删除对象
    batch_delete_objects(&options, objectinfo, &delobj, 0, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("test batch_delete_objects successfully. \n");
    }
    else
    {
        printf("test batch_delete_objects faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
obs_status delete_objects_data_callback(int contentsCount,
    obs_delete_objects *delobjs,
    void *callbackData)
{
    int i;
    for (i = 0; i < contentsCount; i++) {
        const obs_delete_objects*content = &(delobjs[i]);
        printf("delete object result:\nobject key:%s\nerror code:%s\nerror message:%s\ndelete marker:%s
\ndelete marker version_id:%s\n",
                content->key, content->code, content->message, content->delete_marker, content-
>delete_marker_version_id);
    }
    return OBS_STATUS_OK;
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
```



```
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
```



```
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

相关链接

- 关于批量删除对象的API说明，请参见**批量删除对象**。
- 更多关于批量删除对象的代码示例，请参见**Github示例**。
- 批量删除对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

9.10 复制对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

用户可以根据需要将存储在OBS上的对象复制到其他路径下。复制对象操作将创建需要复制的对象的副本，即为指定桶中的对象创建一个副本。在单次操作中，您可以创建最大5GB的对象副本。

接口约束

- 您必须是桶拥有者或拥有复制对象的权限，才能复制对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 复制对象操作的请求需要通过头域携带复制的源桶和对象信息，不能携带消息实体。

- 目标对象大小范围是[0, 5GB]，如果源对象大小超过5GB，只能使用**分段上传-复制段(C SDK)**功能复制部分对象。

方法定义

```
void copy_object(const obs_options *options, char *key, const char *version_id,
    obs_copy_destination_object_info *object_info,
        unsigned int is_copy, obs_put_properties *put_properties,
    server_side_encryption_params *encryption_params,
    obs_response_handler *handler, void *callback_data);
```

请求参数

表 9-212 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
object_info	obs_copy_destination_object_info *	必选	**参数解释：** 指明复制对象信息。 **约束限制：** 无
is_copy	unsigned int	必选	**参数解释：** 用来指定新对象的元数据是从源对象中复制，还是用请求中的元数据替换。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
put_properties	obs_put_properties*	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 服务端加密设置。 **约束限制：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-213 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-214 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 9-215 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-216 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-217 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-218 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-219 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-220 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-221 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-222 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 9-223 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 9-224 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-225 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-226 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-227 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-228 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-229 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-230 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 9-231 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 9-232 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 9-233 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 9-234 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 9-235 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为 application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-236 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-237 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 9-238 obs_copy_destination_object_info

参数名称	参数类型	是否必选	描述
destination_bucket	constchar*	必选	**参数解释：** 复制的目标桶名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
destination_key	constchar*	必选	**参数解释：** 复制的目标对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	constchar*	可选	**参数解释：** 目标对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
last_modified_return	int64_t *	必选	**参数解释：** 对象最近一次被修改的时间（UTC时间）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
etag_return_size	int	必选	**参数解释：** etag_return的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
etag_return	char*	必选	**参数解释：** 新对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无

代码示例一：简单复制

以下示例展示如何通过copy_object来复制对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过copy_object来复制对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    char eTag[OBS_COMMON_LEN_256] = { 0 };
    int64_t lastModified;
    // 设置目的对象信息
    obs_copy_destination_object_info objectinfo = { 0 };
    objectinfo.destination_bucket = "example-dest-bucket-name";
    objectinfo.destination_key = "example_destination_key";
    objectinfo.etag_return = eTag;
    objectinfo.etag_return_size = sizeof(eTag);
    objectinfo.last_modified_return = &lastModified;
    // 设置响应回调函数
    obs_response_handler responseHandler =
    {
```



```
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* source_object_key = "example_source_object_key";
    char* source_object_version_id = NULL;
    // 复制对象
    copy_object(&options, source_object_key, source_object_version_id, &objectinfo, 1, NULL, NULL,
&responseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status && eTag[0] != '\0') {
        printf("test_copy_object successfully. \n");
    }
    else
    {
        printf("test_copy_object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
```



```
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

代码示例二：限定条件复制

说明

- 如果包含if_not_modified_since并且不符合，或者包含if_match_etag并且不符合，或者包含if_modified_since并且不符合，或者包含if_not_match_etag并且不符合，则复制失败，抛出异常中HTTP状态码为：412 precondition failed。
- if_modified_since和if_not_match_etag可以一起使用；if_not_modified_since和if_match_etag可以一起使用。

以下示例展示如何通过copy_object函数进行限定条件复制对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
```



```
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过copy_object函数进行限定条件复制：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    char eTag[OBS_COMMON_LEN_256] = { 0 };
    int64_t lastModified;
    // 设置目的对象信息
    obs_copy_destination_object_info objectinfo = { 0 };
    objectinfo.destination_bucket = "example-dest-bucket-name";
    objectinfo.destination_key = "example_destination_key";
    objectinfo.etag_return = eTag;
    objectinfo.etag_return_size = sizeof(eTag);
    objectinfo.last_modified_return = &lastModified;
    //限定条件
    obs_put_properties putProperties = { 0 };
    init_put_properties(&putProperties);
    obs_get_conditions get_conditions = { 0 };
    init_get_properties(&get_conditions);
    putProperties.get_conditions = &get_conditions;
    putProperties.get_conditions->if_match_etag = "2ea225dbff5b5b1d7bd6bd3b740681cd";
    putProperties.get_conditions->if_modified_since = 0;
    putProperties.get_conditions->if_not_match_etag = "34a7ef8cc629583f3cdd882791f31350";
    putProperties.get_conditions->if_not_modified_since = INT_MAX;
    // 设置响应回调函数
    obs_response_handler responseHandler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* source_object_key = "example_source_object_key";
    char* source_object_version_id = NULL;
    // 复制对象
    copy_object(&options, source_object_key, source_object_version_id, &objectinfo, 1, &putProperties, NULL,
&responseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status && eTag[0] != '\0') {
        printf("test_copy_object successfully. \n");
    }
    else
    {
        printf("test_copy_object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
```



```
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
```



```
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

代码示例三：重写对象访问权限

以下示例展示如何通过copy_object函数进行复制对象时重写复制目的对象的访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过copy_object函数进行复制对象时重写复制目的对象的访问权限
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    char eTag[OBS_COMMON_LEN_256] = { 0 };
    int64_t lastModified;
    // 设置目的对象信息
    obs_copy_destination_object_info objectinfo = { 0 };
    objectinfo.destination_bucket = "example-dest-bucket-name";
    objectinfo.destination_key = "example_destination_key";
    objectinfo.etag_return = eTag;
    objectinfo.etag_return_size = sizeof(eTag);
    objectinfo.last_modified_return = &lastModified;
```



```
    //重写目的对象访问权限
    obs_put_properties putProperties = { 0 };
    init_put_properties(&putProperties);
    putProperties.canned_acl = OBS_CANNED_ACL_PUBLIC_READ;
    // 设置响应回调函数
    obs_response_handler responseHandler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* source_object_key = "example_source_object_key";
    char* source_object_version_id = NULL;
    // 复制对象
    copy_object(&options, source_object_key, source_object_version_id, &objectinfo, 1, &putProperties, NULL,
&responseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status && eTag[0] != '\0') {
        printf("test_copy_object successfully. \n");
    }
    else
    {
        printf("test_copy_object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
```



```
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

相关链接

- 关于复制对象的API说明，请参见**复制对象**。
- 更多关于复制对象的代码示例，请参见**Github示例**。
- 复制对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 复制对象常见问题请参见**我可以在桶间进行文件复制吗？** 。

9.11 重命名对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

重命名对象操作是指将指定并行文件系统内的一个对象重命名为其他对象名。

接口约束

- 您必须是并行文件系统拥有者或拥有重命名文件的权限，才能重命名文件。建议使用IAM或策略进行授权，如果使用IAM则需授予obs:bucket:PutObject权限，如果使用策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的region以及region与endPoint的对应关系，详细信息请参见**地区与终端节点**。
- 目前接口仅在并行文件系统支持，普通对象桶不支持。
- 重命名时，如果源对象不存在，则会报错（HTTP状态码为404）。

方法定义

```
void rename_object(const obs_options *options, char *key, char *new_object_name,
    obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 9-239 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
new_object_name	char *	必选	**参数解释：** 新对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-240 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-241 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-242 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-243 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-244 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-245 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 9-246 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-247 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-248 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-249 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-250 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-251 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 9-252 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-253 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-254 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-255 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：重命名对象

以下示例展示如何通过rename_object函数重命名对象（rename_object接口只适用于并行文件系统，对象桶该接口不支持）：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过rename_object函数重命名对象(rename_object接口只适用于并行文件系统，对象桶该接口不支持)：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "example-posix-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 对象名
    char *key = "example_rename_key_test";
    // 新的对象名
    char *new_key_name = "example_rename_key_test_new";
```



```
    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        // 对象属性可以在response_properties_callback中获取
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 重命名对象
    rename_object(&options, key, new_key_name, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("rename_object successfully. \n");
    }
    else
    {
        printf("rename_object failed.\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
```



```
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

9.12 截断对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以通过调用truncate_object来对对象进行截断操作。截断对象操作是指将指定并行文件系统内的一个对象截断到指定大小。

接口约束

- 如果对象不存在，则会报错（HTTP状态码为404）。
- 截断对象接口只适用于并行文件系统，该接口不支持对象桶。

方法定义

```
void truncate_object(const obs_options *options, char *key, uint64_t object_length,
    obs_response_handler *handler, void *callback_data);
```

参数描述

表 9-256 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
object_length	uint64_t	必选	**参数解释：** 截断到的大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-257 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure *	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 9-258 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 9-259 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 9-260 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-261 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 9-262 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 9-263 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 9-264 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-265 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-266 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 9-267 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-268 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 9-269 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-270 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-271 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 9-272 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：截断对象

以下示例展示如何通过truncate_object函数截断对象（truncate_object接口只适用于并行文件系统，对象桶该接口不支持）：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过truncate_object函数截断对象(truncate_object接口只适用于并行文件系统，对象桶该接口不支持)：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "example-posix-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 对象名
    char *key = "example_truncate_key_test";
    uint64_t object_length = 1024;
    // 设置响应回调函数
```



```
    obs_response_handler response_handler =
    {
        // 对象属性可以在response_properties_callback中获取
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 截断对象
    truncate_object(&options, key, object_length, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("truncate_object successfully. \n");
    }
    else
    {
        printf("truncate_object failed.\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
```



```
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```


10 多段相关接口(C SDK)

10.1 多段相关接口说明(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

对于较大文件上传，可以切分成段上传。用户可以在如下的场景内（但不仅限于此）使用分段上传的模式：

- 上传超过100MB大小的文件。
- 网络条件较差，和OBS服务端之间的连接经常断开。
- 上传前无法确定将要上传文件的大小。

分段上传分为如下3个步骤：

1. **初始化分段上传任务**。
2. **上传段**。
3. **合并段**或**取消分段上传任务**。

分段上传的主要目的是解决大文件上传或网络条件较差的情况。下面的代码示例展示了如何使用分段上传并发上传大文件：

```
static void test_concurrent_upload_part(char *filename, char *key, uint64_t uploadSliceSize)
{
    obs_status ret_status = OBS_STATUS_BUTT;
    // 创建并初始化option
    obs_options option;
    init_obs_options(&option);
    option.bucket_options.host_name = "<your-endpoint>";
    option.bucket_options.bucket_name = "<Your bucketname>";

    //从环境变量读取ak/sk
    option.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    option.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char *concurrent_upload_id;
    uint64_t uploadSize = uploadSliceSize;
    uint64_t filesize =0;
```

```
    //初始化结构体put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //大文件信息:文件指针，文件大小，按照分段大小的分段数
    test_upload_file_callback_data data;
    memset(&data, 0, sizeof(test_upload_file_callback_data));
    filesize = get_file_info(filename,&data);
    data.noStatus = 1;
    data.part_size = uploadSize;
    data.part_num = (filesize % uploadSize == 0) ? (filesize / uploadSize) : (filesize / uploadSize +1);
    // 初始化上传段回调函数
    obs_response_handler Handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    // 合并段回调函数
    obs_complete_multi_part_upload_handler complete_multi_handler =
    {
       {&response_properties_callback,
        &response_complete_callback},
       &CompleteMultipartUploadCallback
    };
    //初始化上传段任务返回uploadId: uploadIdReturn
    char uploadIdReturn[256] = {0};
    int upload_id_return_size = 255;
    initiate_multi_part_upload(&option,key,upload_id_return_size,uploadIdReturn, &putProperties,
           0,&Handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
       printf("test init upload part return uploadIdReturn(%s). \n", uploadIdReturn);
       strcpy(concurrent_upload_id,uploadIdReturn);
    }
    else
    {
       printf("test init upload part faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 并发上传
    test_concurrent_upload_file_callback_data *concurrent_upload_file;
    concurrent_upload_file =(test_concurrent_upload_file_callback_data *)malloc(
                  sizeof(test_concurrent_upload_file_callback_data)*(data.part_num+1));
    if(concurrent_upload_file == NULL)
    {
        printf("malloc test_concurrent_upload_file_callback_data failed!!!");
        return ;
    }
    test_concurrent_upload_file_callback_data *concurrent_upload_file_complete =
                           concurrent_upload_file;
    start_upload_threads(data, concurrent_upload_id,filesize, key, option, concurrent_upload_file_complete);
     // 合并段
    obs_complete_upload_Info *upload_Info = (obs_complete_upload_Info *)malloc(
              sizeof(obs_complete_upload_Info)*data.part_num);
    for(i=0; i<data.part_num; i++)
    {
       upload_Info[i].part_number = concurrent_upload_file_complete[i].part_num;
       upload_Info[i].etag = concurrent_upload_file_complete[i].etag;
    }
    complete_multi_part_upload(&option, key, uploadIdReturn,
data.part_num,upload_Info,&putProperties,&complete_multi_handler,&ret_status);
    if (ret_status == OBS_STATUS_OK) {
       printf("test complete upload successfully. \n");
    }
    else
    {
       printf("test complete upload faied(%s).\n", obs_get_status_name(ret_status));
    }
    if(concurrent_upload_file)
    {
       free(concurrent_upload_file);
       concurrent_upload_file = NULL;
    }
```



```
    if(upload_Info)
    {
        free(upload_Info);
        upload_Info = NULL;
    }
}
```

说明

大文件分段上传时，使用**Offset参数**和**PartSize参数**配合指定每段数据在文件中的起始结束位置。

注意

并发数过大，可能会因为网络不稳定等原因，产生Timeout错误，需要限制并发数。

10.2 分段上传-初始化分段上传任务(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

使用多段上传特性时，用户必须首先调用创建多段上传任务接口创建任务，系统会给用户返回一个全局唯一的多段上传任务号，作为任务标识。后续用户可以根据这个标识发起相关的请求，如：上传段、合并段、列举段等。创建多段上传任务不影响已有的同名对象；同一个对象可以同时存在多个多段上传任务；每个多段上传任务在初始化时可以附加消息头信息，包括acl、用户自定义元数据和通用的HTTP消息头contentType、contentEncoding等，这些附加的消息头信息将先记录在多段上传任务元数据中。

在指定桶中初始化分段上传任务。

接口约束

- 您必须是桶拥有者或拥有初始化分段上传任务的权限，才能初始化分段上传任务。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 初始化上传段任务并上传一个或多个段之后，您必须**合并段**或**取消多段上传任务**，才能停止收取已上传的段的存储费用。仅当在合并段或取消多段上传任务之后，OBS才释放段存储并停止向您收取段存储费用。

方法定义

```
void initiate_multi_part_upload(const obs_options *options, char *key,int upload_id_return_size,
    char *upload_id_return, obs_put_properties *put_properties,
```

```
    server_side_encryption_params *encryption_params,
    obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 10-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
upload_id_return_size	int	必选	**参数解释：** 多段上传ID缓存大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
upload_id_return	char *	必选	**参数解释：** 多段上传ID缓存。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
put_properties	**obs_put_properties***	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 请详见**创建访问密钥**。。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 10-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-5 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-6 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-7 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 10-8 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-11 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD %E6%96%87: %E4%B8%AD %E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 10-12 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 10-13 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 10-14 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 10-15 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为 application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-16 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-17 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 10-18 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 10-19 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 10-20 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback*	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-21 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-22 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 10-23 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-24 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-25 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-26 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：初始化分段上传任务

以下示例展示如何通过initiate_multi_part_upload初始化一个分段上传任务：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何通过initiate_multi_part_upload初始化一个分段上传任务：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 定义多段上传id缓存及大小
    char upload_id[OBS_COMMON_LEN_256] = { 0 };
    int upload_id_size = OBS_COMMON_LEN_256;
    // 设置响应回调函数
    obs_response_handler handler =
    {
```



```
    &response_properties_callback,
    &response_complete_callback
};
// 分段上传对象名
char *key = "example_multi_part_upload_object_key";
// 初始化分段上传任务,这里的upload_id就是接口定义中的upload_id_return
obs_status ret_status;
initiate_multi_part_upload(&options, key, upload_id_size, upload_id, NULL, NULL, &handler, &ret_status);
if (OBS_STATUS_OK == ret_status)
{
    printf("test init upload part successfully. uploadId= %s\n", upload_id);
}
else
{
    printf("test init upload part faied(%s).\n", obs_get_status_name(ret_status));
}
// 释放分配的全局资源
obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
if (properties == NULL)
{
    printf("error! obs_response_properties is null!");
    if (callback_data != NULL)
    {
        obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
        printf("server_callback buf is %s, len is %llu",
            data->buffer, data->buffer_len);
        return OBS_STATUS_OK;
    }
    else {
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                              \
do {                                                            \
    if (properties-> field) {                                   \
        printf("%s: %s\n", name, properties->field);          \
    }                                                           \
} while (0)
print_nonnull("request_id", request_id);
print_nonnull("request_id2", request_id2);
print_nonnull("content_type", content_type);
if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
}
print_nonnull("server", server);
print_nonnull("ETag", etag);
print_nonnull("expiration", expiration);
print_nonnull("website_redirect_location", website_redirect_location);
print_nonnull("version_id", version_id);
print_nonnull("allow_origin", allow_origin);
print_nonnull("allow_headers", allow_headers);
print_nonnull("max_age", max_age);
print_nonnull("allow_methods", allow_methods);
print_nonnull("expose_headers", expose_headers);
print_nonnull("storage_class", storage_class);
print_nonnull("server_side_encryption", server_side_encryption);
print_nonnull("kms_key_id", kms_key_id);
print_nonnull("customer_algorithm", customer_algorithm);
print_nonnull("customer_key_md5", customer_key_md5);
print_nonnull("bucket_location", bucket_location);
print_nonnull("obs_version", obs_version);
print_nonnull("restore", restore);
print_nonnull("obs_object_type", obs_object_type);
```



```
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于分段上传-初始化分段上传任务的API说明，请参见**初始化上传段任务**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.3 分段上传-上传段(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

初始化分段上传任务后，通过分段上传任务的ID，上传段到指定桶中。除了最后一段以外，其他段的大小范围是100KB~5GB；最后一段的大小范围是0~5GB。上传的段的编号也有范围限制，其范围是1~10000。

上传段时，除了指定上传ID，还必须指定段编号。您可以选择1和10000之间的任意段编号。段编号在您正在上传的对象中唯一地标示了段及其位置。如果您使用之前上传的段的同一段编号上传新段，则之前上传的段将被覆盖。无论您何时上传段，OBS都将在其响应中返回ETag标头。对于每个段上传任务，您必须记录每个段编号和ETag值。您在随后的请求中需要添加这些值以完成多段上传。

接口约束

- 您必须是桶拥有者或拥有上传段的权限，才能上传段。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 初始化上传段任务并上传一个或多个段之后，您必须合并段或取消多段上传任务，才能停止收取已上传的段的存储费用。仅当在合并段或取消多段上传任务之后，OBS才释放段存储并停止向您收取段存储费用。
- 段任务中的partNumber是唯一的，重复上传相同partNumber的段，后一次上传会覆盖前一次上传内容。多并发上传同一对象的同一partNumber时，服务端遵循Last Write Win策略，但“Last Write”的时间定义为段元数据创建时间。为了保证数据准确性，客户端需要加锁保证同一对象的同一个段上传的并发性。同一对象的不同段并发上传不需要加锁。
- 上传段接口要求除最后一段以外，其他的段大小都要大于100KB。但是上传段接口并不会立即校验上传段的大小（因为不知道是否为最后一块）；只有调用合并段接口时才会校验。
- OBS会将服务端收到段数据的ETag值（段数据的MD5值）返回给用户。
- 为了保证数据在网络传输过程中不出现错误，可以通过设置MD5值，并放到Content-MD5请求头中；OBS服务端会计算上传数据的MD5值与SDK计算的MD5值比较，保证数据完整性。
- 分段号的范围是1~10000。如果超出这个范围，OBS将返回400 Bad Request错误。
- OBS 3.0的桶支持最小段的大小为100KB，OBS 2.0的桶支持最小段的大小为5MB。请在OBS 3.0的桶上执行分段上传操作。

方法定义

```
void upload_part(const obs_options *options, char *key, obs_upload_part_info *upload_part_info,
    uint64_t content_length, obs_put_properties *put_properties,
    server_side_encryption_params *encryption_params,
    obs_upload_handler *handler, void *callback_data);
```

请求参数说明

表 10-27 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
upload_part_info	**obs_upload_part_info** *	必选	**参数解释：** 上传段的信息。 **约束限制：** 无
content_length	uint64_t	必选	**参数解释：** 对象内容长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 如果不设置，则SDK会自动计算对象数据的长度。

参数名称	参数类型	是否必选	描述
put_properties	obs_put_properties*	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 上传对象加密设置。 **约束限制：** 无
handler	obs_upload_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-28 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-29 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 10-30 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-31 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-32 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-33 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 10-34 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-35 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-36 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-37 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： • kms • AES256 **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 10-38 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 10-39 obs_upload_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
upload_data_callback	**obs_upload_data_callback** *	必选	**参数解释：** 回调函数指针，用于把要上传的数据复制到待上传的数据缓冲区。 **约束限制：** 无
progress_callback	**obs_progress_callback_internal** *	必选	**参数解释：** 进度回调函数指针。 **约束限制：** 无

表 10-40 obs_progress_callback_internal

参数名称	参数类型	是否必选	描述
now	uint64_t	必选	**参数解释：** 已上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
total	uint64_t	必选	**参数解释：** 总共需要上传的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-41 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-42 obs_response_properties_callback

函数参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-43 obs_response_complete_callback

函数参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

函数参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-44 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-45 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-46 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-47 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 10-48 obs_upload_part_info

参数名称	参数类型	是否必选	描述
part_number	unsigned int	必选	**参数解释:** 上传段的段号。 **约束限制:** 无 **取值范围:** 从1到10000的整数。 **默认取值:** 无
upload_id	int	必选	**参数解释:** 多段上传任务ID。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 10-49 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
get_conditions	obs_get_conditions *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	**obs_name_value***	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	**metadata_action_indicator**	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见**metadata_action_indicator**。

参数名称	参数类型	是否必选	描述
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 10-50 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 10-51 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 10-52 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 10-53 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-54 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-55 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。

枚举值	说明
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 10-56 obs_upload_data_callback

函数参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：上传段

以下示例展示如何通过upload_part上传段：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
```



```
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct test_upload_file_callback_data
{
    obs_status ret_status;
    FILE *infile;
    int part_num;
    uint64_t part_size;
    uint64_t start_byte;
} test_upload_file_callback_data;
uint64_t get_file_info(char *localfile, test_upload_file_callback_data *data);
int test_upload_file_data_callback(int buffer_size, char *buffer, void *callback_data);
int main()
{
    // 以下示例展示如何通过upload_part上传段：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 分段上传对象名
    char *key = "example_multi_part_upload_object_key";
    // 分段上传的任务ID
    char* uploadID = "00000194178F94B64016E9C3BD03E89B";
    // 定义分片大小5M
    uint64_t uploadSliceSize = 5L * 1024 * 1024;
    // 上传的文件名
    char file_name[256] = "./example_local_file_to_upload.tar";
    // 定义并初始化上传段大小
    uint64_t uploadSize = uploadSliceSize;
    // 定义并初始化上传文件长度变量
    uint64_t fileSize = 0;
    //初始化put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    //回调函数
    obs_upload_handler handler =
    {
        {&response_properties_callback, &response_complete_callback},
        &test_upload_file_data_callback
    };
    //回调数据初始化
    test_upload_file_callback_data data;
    memset(&data, 0, sizeof(test_upload_file_callback_data));
    // 因为要上传两段，文件大小需要大于uploadSliceSize
    fileSize = get_file_info(file_name,&data);
    if (fileSize == 0) {
        // 打开文件失败
        return -1;
    }
    data.part_size = uploadSize;
    data.part_num = (fileSize % uploadSize == 0) ? (fileSize / uploadSize) : (fileSize / uploadSize +1);
    obs_upload_part_info uploadPartInfo;
    memset(&uploadPartInfo, 0, sizeof(obs_upload_part_info));
```



```
    //上传第一段
    uploadPartInfo.part_number = 1;
    uploadPartInfo.upload_id = uploadID;
    data.start_byte  = 0;
    data.ret_status = OBS_STATUS_BUTT;
    upload_part(&options,key,&uploadPartInfo,uploadSize,
                                &put_properties,0,&handler,&data);
    if (data.infile != NULL) {
        fclose(data.infile);
        data.infile = NULL;
    }
    if (OBS_STATUS_OK == data.ret_status) {
        printf("test upload part 1 successfully. \n");
    }
    else
    {
        printf("test upload part 1 faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    //上传第二段
    uploadPartInfo.part_number = 2;
    uploadPartInfo.upload_id = uploadID;
    fileSize = get_file_info(file_name, &data);
    uploadSize = fileSize - uploadSize;
    data.part_size = uploadSize;
    data.start_byte = uploadSliceSize;
    data.ret_status = OBS_STATUS_BUTT;
    upload_part(&options,key,&uploadPartInfo,uploadSize, &put_properties,0,&handler,&data);
    if (data.infile != NULL) {
        fclose(data.infile);
        data.infile = NULL;
    }
    if (OBS_STATUS_OK == data.ret_status) {
        printf("test upload part 2 successfully. \n");
    }
    else
    {
        printf("test upload part 2 faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
```



```
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        test_upload_file_callback_data *data =
            (test_upload_file_callback_data *)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
int test_upload_file_data_callback(int buffer_size, char *buffer, void *callback_data)
```



```
{
    test_upload_file_callback_data *data =
        (test_upload_file_callback_data *)callback_data;
    int ret = 0;
    fseek(data->infile, data->start_byte, SEEK_SET);
    int toRead = ((data->part_size > (unsigned)buffer_size) ?
        (unsigned)buffer_size : data->part_size);
    ret = fread(buffer, 1, toRead, data->infile);
    data->start_byte += ret;
    return ret;
}
uint64_t get_file_info(char *localfile, test_upload_file_callback_data *data)
{
    data->infile = 0;
    uint64_t content_length = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    return content_length;
}
```

相关链接

- 关于分段上传-上传段的API说明，请参见**上传段**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.4 分段上传-合并段(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

如果用户上传完所有的段，就可以调用合并段接口，系统将在服务端将用户指定的段合并成一个完整的对象。在执行“合并段”操作以前，用户不能下载已经上传的数据。在合并段时需要将多段上传任务初始化时记录的附加消息头信息复制到对象元数据中，其处理过程和普通上传对象带这些消息头的处理过程相同。在并发合并段的情况下，仍然遵循Last Write Win策略，但“Last Write”的时间定义为段任务的初始化时间。

已经上传的段，只要没有取消对应的多段上传任务，都要占用用户的容量配额；对应的多段上传任务“合并段”操作完成后，只有指定的多段数据占用容量配额，用户上

传的其他此多段任务对应的段数据如果没有包含在“合并段”操作指定的段列表中，“合并段”完成后系统将删除多余的段数据，且同时释放容量配额。

合并段时，OBS通过按升序的段编号规范化多段来创建对象。如果在初始化上传段任务中提供了任何对象元数据，则OBS会将该元数据与对象相关联。成功完成请求后，段将不再存在。合并段请求必须包括上传ID以及段编号和相应的ETag值的列表。OBS响应包括可唯一地识别组合对象数据的ETag。此ETag无需成为对象数据的MD5哈希。

接口约束

- 您必须是桶拥有者或拥有合并段的权限，才能合并段。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 如果上传了10个段，但合并时只选择了9个段进行合并，那么未被合并的段将会被系统自动删除，未被合并的段删除后不能恢复。在进行合并之前请使用列出已上传的段接口进行查询，仔细核对所有段，确保没有段被遗漏。

方法定义

```
void complete_multi_part_upload(const obs_options *options, char *key, const char *upload_id,
    unsigned int part_number,
    obs_complete_upload_Info *complete_upload_Info, obs_put_properties *put_properties,
    obs_complete_multi_part_upload_handler *handler, void *callback_data);
```

请求参数说明

表 10-57 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
upload_id	char *	必选	**参数解释：** 指明多段上传任务。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
part_number	unsigned int	必选	**参数解释：** 段个数，complete_upload_Info数组长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
complete_upload_Info	**obs_complete_upload_Info** *	必选	**参数解释：** 段信息数组。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
put_properties	obs_put_properties*	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
handler	obs_complete_multi_part_upload_handler*	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-58 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-59 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 10-60 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-61 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-62 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-63 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 10-64 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-65 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 10-66 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-67 obs_complete_multi_part_upload_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
complete_multipart_upload_callback	**obs_complete_multi_part_upload_callback** *	必选	**参数解释：** 合并段回调函数指针，用于获取合并段相关信息。 **约束限制：** 无

表 10-68 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback*	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-69 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-70 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-71 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-72 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-73 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-74 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 10-75 obs_complete_upload_Info

参数名称	参数类型	是否必选	描述
part_number	unsigned int	必选	**参数解释：** 分段的段号。 **约束限制：** 无 **取值范围：** 从1到10000的整数。 **默认取值：** 无
etag	char *	必选	**参数解释：** 对应段的ETag值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-76 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
get_conditions	obs_get_conditions *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** • 取值必须大于0。 • 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	**obs_name_value***	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	**metadata_action_indicator**	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见**metadata_action_indicator**。

参数名称	参数类型	是否必选	描述
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 10-77 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 10-78 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 10-79 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 10-80 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-81 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	**image_process_mode_type**	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见**image_process_mode_type**。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-82 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。

枚举值	说明
obs_image_process_style	图片处理参数以style开头。

表 10-83 obs_complete_multi_part_upload_callback

参数名称	参数类型	是否必选	描述
location	const char *	必选	**参数解释：** 合并后得到对象的路径。 **约束限制：** 格式：/bucketName/objectName **取值范围：** 无 **默认取值：** 无
bucket	const char *	必选	**参数解释：** 合并段所在的桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
key	const char *	必选	**参数解释：** 合并段后得到的对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
etag	const char *	必选	**参数解释：** 合并段后根据各个段的ETag值计算出的结果，是对象内容的唯一标识。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 格式：/bucketName/objectName **取值范围：** 无 **默认取值：** 无

代码示例：合并段

以下示例展示如何通过complete_multi_part_upload合并段：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
```



```
obs_status CompleteMultipartUploadCallback_Intern(const char *location,
    const char *bucket,
    const char *key,
    const char *etag,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过complete_multi_part_upload合并段：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;

    // 初始化结构体put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 要合并的分段上传对象名
    char *key = "example_multi_part_upload_object_key";
    // 设置分段信息
    char *uploadId = "000001930065F7C64014C38C138D654B";
    obs_complete_upload_Info info[2];
    info[0].part_number = 1;
    info[0].etag = "4662995f27b37f9e3cdd42f682f29b27";
    info[1].part_number = 2;
    info[1].etag = "cd37aa94cf614fe5cded93707cbf7f88";
    unsigned int part_count = sizeof(info) / sizeof(obs_complete_upload_Info);
    // 设置响应回调函数
    obs_complete_multi_part_upload_handler Handler =
    {
        {&response_properties_callback,
         &response_complete_callback},
        &CompleteMultipartUploadCallback_Intern
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 合并段
    complete_multi_part_upload(&options, key, uploadId, part_count, info, &put_properties,
        &Handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("test complete upload successfully. \n");
    }
    else
    {
        printf("test complete upload faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
```



```
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
```



```
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
obs_status CompleteMultipartUploadCallback_Intern(const char *location,
    const char *bucket,
    const char *key,
    const char *etag,
    void *callback_data)
{
    printf("location = %s \nbucket = %s \nkey = %s \neTag = %s \n", location, bucket, key, etag);
    return OBS_STATUS_OK;
}
```

说明

- 上面代码中的info结构体数组是进行上传段后保存的分段号和分段ETag值的列表。
- 分段可以是不连续的。

相关链接

- 关于分段上传-合并段的API说明，请参见**合并段**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.5 分段上传-列举分段上传任务(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

列举指定桶中所有的初始化后还未合并或还未取消的分段上传任务。

通过列举桶中的多段上传任务，您可以获得已初始化多段上传任务的列表，已初始化多段上传任务是指初始化后还未合并以及未取消的多段上传任务。每个请求将返回最多1000个多段上传任务，如果正在进行的多段上传任务超过1000个，您需要发送其他请求以检索剩余的多段上传任务。

接口约束

- 您必须是桶拥有者或拥有列举分段上传任务的权限，才能列举分段上传任务。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:ListBucketMultipartUploads权限，如果使用桶策略则需授予ListBucketMultipartUploads权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 您必须得到可以对桶执行ListBucketMultipartUploads操作的权限，才能列出正在上传到该桶的多段上传。
- 除了默认情况之外，桶所有者可以允许其他委托人对桶执行ListBucketMultipartUploads操作。

方法定义

```
void list_multipart_uploads(const obs_options *options, const char *prefix, const char *marker,
        const char *delimiter, const char* uploadid_marker, int max_uploads,
        obs_list_multipart_uploads_handler *handler, void *callback_data);
```

请求参数说明

表 10-84 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
prefix	const char *	必选	**参数解释：** 限定返回的分段上传任务中的对象名必须带有prefix前缀。 例如，假设您拥有以下对象：logs/day1、logs/day2、logs/day3和ExampleObject.jpg。如果您将logs/指定为前缀，将返回以字符串“logs/”开头的三个对象所在的分段上传任务。如果您指定空的前缀且请求中没有其他过滤条件，将返回桶中的所有分段上传任务。 **约束限制：** 长度大于0且不超过1024的字符串。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
marker	const char *	必选	**参数解释：** 列举分段上传任务的起始位置。表示列举时返回指定的keyMarker之后的分段上传任务。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
delimiter	const char *	必选	**参数解释：** 对分段上传任务中的对象名进行分组的字符。通常与前缀prefix搭配使用，如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条commonPrefixes；如果没有指定prefix，从对象名的首字符到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条commonPrefixes。 例如，桶中有3个对象，分别为abcd、abcde、bbcde。如果指定delimiter为d，prefix为a，abcd、abcde会被分成一组，形成一条前缀为abcd的commonPrefixes；如果只指定delimiter为d，abcd、abcde会被分成一组，形成一条前缀为abcd的commonPrefixes，而bbcde会被单独分成一组，形成一条前缀为bbcd的commonPrefixes。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
uploadid_marker	const char *	必选	**参数解释：** 列举分段上传任务的起始位置（uploadId标识）。 **约束限制：** 只有与keyMarker参数一起使用时才有意义，即列举时返回指定keyMarker的uploadIdMarker之后的分段上传任务。 **取值范围：** 无 **默认取值：** 无
max_uploads	int	必选	**参数解释：** 列举分段上传任务的最大数目。 **约束限制：** 当该参数超出1000时，按照默认的1000进行处理。 **取值范围：** [1，1000]，单位：个。 **默认取值：** 1000
handler	**obs_list_multipart_uploads_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-85 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-86 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 10-87 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-88 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-89 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-90 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 10-91 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-92 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-93 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-94 obs_list_multipart_uploads_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
list_mulpu_callback	**obs_list_multipart_uploads_callback** *	必选	**参数解释：** 回调函数指针，用于把列举出的数据复制到用户自定义回调数据中。 **约束限制：** 无

表 10-95 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback*	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-96 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 10-97 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-98 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-99 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-100 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-101 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 10-102 obs_list_multipart_uploads_callback

参数名称	参数类型	是否必选	描述
is_truncated	int	必选	**参数解释：** 表明本次返回的结果列表是否被截断。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
next_marker	const char *	必选	**参数解释：** 如果本次没有返回全部结果，标明接下来请求的marker值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
next_uploadId_marker	const char *	必选	**参数解释：** 如果本次没有返回全部结果，标明接下来请求的uploadid_marker值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
uploads_count	int	必选	**参数解释**： uploads的数量。 **约束限制：** 该字段只适用于多版本列举场景。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
uploads	const **obs_list_multipart_upload** *	必选	**参数解释：** 列举出的多版本相关信息。 **约束限制：** 无
common_prefixes_count	int	必选	**参数解释：** common_prefixes的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
common_prefixes	const char **	必选	**参数解释：** 当请求中设置了delimiter分组字符时，返回按delimiter分组后的对象名称前缀列表。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-103 obs_list_multipart_upload

参数名称	参数类型	是否必选	描述
key	const char *	可选	**参数解释：** 对象名。对象名是对象在存储桶中的唯一标识。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
upload_id	const char *	可选	**参数解释：** 分段上传任务的ID。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
initiator_id	const char *	可选	**参数解释**： 分段上传任务的创建者。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
owner_id	const char *	可选	**参数解释：** 对象拥有者的domain_id（账号ID）。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
initiated	int64_t	可选	**参数解释：** 分段上传任务的初始化时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：列举分段上传任务

以下示例展示如何通过list_multipart_uploads列举分段上传任务：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
// 列举分段上传任务回调函数，可以在这个回调中把列举出的分段上传任务的相关信息内容记录到callback_data(用户自定义回调数据)中
obs_status list_multipart_uploads_callback(int is_truncated, const char *next_marker,
    const char *next_uploadId_marker, int uploads_count,
    const obs_list_multipart_upload *uploads, int common_prefixes_count,
    const char **common_prefixes, void *callback_data);
int main()
{
    // 以下示例展示如何通过list_multipart_uploads列举分段上传任务：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 列举对象名以prefix开始的任务信息
    char *prefix = "example_multi_part_upload_object_key";
    // 列举时返回指定的key-marker之后的多段任务。
    char *marker = "";
    // 用于分组列举的结果
    char *delimiter = "/";
    // 只有和key-marker一起使用才有意义， 列举时返回指定的key-marker的upload-id-marker之后的多段任务。
    char *uploadid_marker = "";
    // 列举的多段任务的最大条目，取值范围为[1, 1000]，当超出范围时，按照默认的1000进行处理。
    int max_uploads = 1000;
    // 设置响应回调函数
```



```
    obs_list_multipart_uploads_handler handler = {
        {
            &response_properties_callback,
            &response_complete_callback
        },
        &list_multipart_uploads_callback
    };
    // 列举分段上传任务
    obs_status ret_status;
    list_multipart_uploads(&options, prefix, marker, delimiter, NULL, NULL, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("test list_multipart_uploads successfully.\n");
    }
    else
    {
        printf("test list_multipart_uploads faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 列举分段上传任务回调函数，可以在这个回调中把列举出的分段上传任务的相关信息内容记录到
callback_data(用户自定义回调数据)中
obs_status list_multipart_uploads_callback(int is_truncated, const char *next_marker,
    const char *next_uploadId_marker, int uploads_count,
    const obs_list_multipart_upload *uploads, int common_prefixes_count,
    const char **common_prefixes, void *callback_data) {
    printf("is_truncated ? %s\n", is_truncated ? "yes" : "no");
    printf("next_marker is %s\n", next_marker);
    printf("next_uploadId_marker is %s\n", next_uploadId_marker);
    for (int i = 0; i < uploads_count; ++i) {
        printf("multipart upload task %d. object key is %s\n", i, uploads[i].key);
        printf("multipart upload task %d. object upload_id is %s\n", i, uploads[i].upload_id);
        printf("multipart upload task %d. object initiator_id is %s\n", i, uploads[i].initiator_id);
        printf("multipart upload task %d. object initiator_display_name is %s\n", i,
uploads[i].initiator_display_name);
        printf("multipart upload task %d. object owner_id is %s\n", i, uploads[i].owner_id);
        printf("multipart upload task %d. object owner_display_name is %s\n", i,
uploads[i].owner_display_name);
        printf("multipart upload task %d. object storage_class is %s\n", i, uploads[i].storage_class);
        printf("multipart upload task %d. object initiated is %lld\n", i, uploads[i].initiated);
    }
    for (int i = 0; i < common_prefixes_count; ++i) {
        printf("common_prefixes %d is %s\n", i, common_prefixes[i]);
    }
    return OBS_STATUS_OK;
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
```

```
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
```



```
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于分段上传-列举分段上传任务的API说明，请参见**列举桶中已初始化多段任务**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.6 分段上传-列举已上传的段(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

通过分段上传任务的ID，列举指定桶中已上传的段。

您可以列出特定多段上传任务或所有正在进行的多段上传任务的分段。列举已上传的段操作将返回您已为特定多段上传任务而上传的段信息。对于每个列举已上传的段请求，OBS将返回有关特定多段上传任务的分段信息，最多为1000个分段。如果多段上传中的段超过1000个，您必须发送一系列列举已上传的段请求以检索所有段。请注意，返回的分段列表不包括已合并的分段。

接口约束

- 您必须是桶拥有者或拥有列举已上传的段的权限，才能列举已上传的段。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:ListMultipartUploadParts权限，如果使用桶策略则需授予ListMultipartUploadParts权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void list_parts (const obs_options *options, char *key, list_part_info *listpart,
        obs_list_parts_handler *handler, void *callback_data);
```

请求参数说明

表 10-104 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	constchar *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
listpart	**list_part_info** *	必选	**参数解释：** 指明多段上传任务。 **约束限制：** 无
handler	**obs_list_parts_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-105 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-106 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 10-107 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-108 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-109 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-110 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 10-111 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-112 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-113 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-114 list_part_info

参数名称	参数类型	是否必选	描述
upload_id	char *	可选	**参数解释：** 分段上传任务的ID。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
max_parts	unsigned int	可选	**参数解释**: 规定在列举已上传段响应中的最大Part数目。 **约束限制**: 无 **取值范围**: 无 **默认取值**: 无
part_number_marker	unsigned int	可选	**参数解释**: 指定List的起始位置，只有Part Number数目大于该参数的Part会被列出。 **约束限制**: 无 **取值范围**: 长度为32的字符串。 **默认取值**: 无

表 10-115 obs_list_parts_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释**: 响应回调函数结构体。 **约束限制**: 无
list_parts_callback_ex	**obs_list_parts_callback_ex** *	必选	**参数解释**: 列举段回调，用于获取列举出的分段信息。 **约束限制**: 无

表 10-116 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-117 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-118 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 10-119 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** 请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-120 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-121 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-122 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 10-123 obs_list_parts_callback_ex

参数名称	参数类型	是否必选	描述
uploaded_parts	**obs_uploaded_parts_total_info** *	必选	**参数解释：** 表明分段任务相关信息。 **约束限制：** 无
parts	**obs_list_parts** *	必选	**参数解释：** 分段信息数组。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-124 obs_uploaded_parts_total_info

参数名称	参数类型	是否必选	描述
is_truncated	int	必选	**参数解释:** 表明本次返回的结果列表是否被截断。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
nextpart_number_marker	unsigned int	必选	**参数解释:** 如果本次没有返回全部结果，标明接下来请求的part_number_marker值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
initiator_id	char *	必选	**参数解释:** 分段上传任务的创建者的domain_id（账号ID）。 **约束限制:** 无 **取值范围:** 如何获取账号ID请参见**如何获取账号ID和用户ID?** **默认取值:** 无
owner_id	char *	必选	**参数解释**: 和initiator_id相同。 **约束限制:** 无 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
storage_class	char *	必选	**参数解释:** 存储类别。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
parts_count	int	必选	**参数解释:** **obs_list_parts**的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 10-125 obs_list_parts

参数名称	参数类型	是否必选	描述
part_number	unsigned int	可选	**参数解释:** 已上传段的编号。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
last_modified	int64_t	可选	**参数解释**: 段上传的时间。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
etag	const char *	可选	**参数解释**: 已上传段内容的ETag，是段内容的唯一标识，用于段合并时校验数据一致性。 **约束限制:** 无 **取值范围:** 长度为32的字符串。 **默认取值:** 无
size	uint64_t	必选	**参数解释**: 已上传段大小。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
storage_class	const char *	可选	**参数解释**: 存储类别。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

代码示例：列举已上传的段

以下示例展示如何通过list_parts列举已上传的段：

```
#include "eSDKOBS.h"
#include <stdio.h>
```



```
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
// 列举分段回调函数，可以在这个回调中把列举出的分段的相关信息内容记录到callback_data(用户自定义回调数据)中
obs_status list_parts_callback(obs_uploaded_parts_total_info* uploaded_parts,
    obs_list_parts *parts, void *callback_data);
int main()
{
    // 以下示例展示如何通过list_parts列举已上传的段：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 列举分段的对象名
    char *key = "example_multi_part_upload_object_key";
    list_part_info list_part_infos = { 0 };
    list_part_infos.upload_id = "00000194178F94B64016E9C3BD03E89B";
    list_part_infos.max_parts = 1000;
    list_part_infos.part_number_marker = 0;
    // 设置响应回调函数
    obs_list_parts_handler handler = {
        {
            &response_properties_callback,
            &response_complete_callback
        },
        &list_parts_callback
    };
    obs_status ret_status;
    list_parts(&options, key, &list_part_infos, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("test list_parts successfully.\n");
    }
    else
    {
        printf("test list_parts faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 列举分段回调函数，可以在这个回调中把列举出的分段的相关信息内容记录到callback_data(用户自定义回调数据)中
obs_status list_parts_callback(obs_uploaded_parts_total_info* uploaded_parts,
    obs_list_parts *parts, void *callback_data) {
    if (uploaded_parts == NULL) {
        printf("uploaded_parts is NULL.\n");
    }
    else {
        printf("uploaded_parts->initiator_id is %s\n", uploaded_parts->initiator_id);
        printf("uploaded_parts->initiator_display_name is %s\n", uploaded_parts->initiator_display_name);
        printf("uploaded_parts->is_truncated is %d\n", uploaded_parts->is_truncated);
        printf("uploaded_parts->nextpart_number_marker is %u\n", uploaded_parts-
```



```
>nextpart_number_marker);
        printf("uploaded_parts->owner_id is %s\n", uploaded_parts->owner_id);
        printf("uploaded_parts->owner_display_name is %s\n", uploaded_parts->owner_display_name);
        printf("uploaded_parts->storage_class is %s\n", uploaded_parts->storage_class);
        printf("uploaded_parts->parts_count is %d\n", uploaded_parts->parts_count);
        if (parts == NULL) {
            printf("parts is NULL.\n");
        }
        else {
            for (int i = 0; i < uploaded_parts->parts_count; ++i) {
                printf("parts%d->part_number is %u\n", i + 1, (parts + i)->part_number);
                printf("parts%d->last_modified is %lld\n", i + 1, (parts + i)->last_modified);
                printf("parts%d->etag is %s\n", i + 1, (parts + i)->etag);
                printf("parts%d->size is %llu\n", i + 1, (parts + i)->size);
                printf("parts%d->storage_class is %s\n", i + 1, (parts + i)->storage_class);
            }
        }
    }
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
```



```
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于分段上传-列举已上传的段的API说明，请参见**列举已上传的段**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.7 分段上传-复制段(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

初始化分段上传任务后，通过分段上传任务的ID，复制段到指定桶中。

多段上传任务创建后，用户可以通过指定多段上传任务号，为特定的任务上传段。添加段的方式还包括调用段复制接口。允许用户将已上传对象的一部分或全部复制为段。

将源对象object复制为一个段part1，如果在复制操作之前part1已经存在，复制操作执行之后，旧的段数据part1会被新复制的段数据覆盖。复制成功后，只能列举到最新的段part1，旧的段数据将会被删除。因此在使用复制段接口时请确保目标段不存在或者已无价值，避免因复制段导致数据误删除。复制过程中源对象object无任何变化。

接口约束

- 您必须是桶拥有者或拥有复制段的权限，才能复制段。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 复制段的结果不能仅根据HTTP返回头域中的status_code来判断请求是否成功，头域中status_code返回200时表示服务端已经收到请求，且开始处理复制段请求。复制是否成功会在响应消息的body中，只有body体中有ETag标签才表示成功，否则表示复制失败。

方法定义

```
void copy_part(const obs_options *options, char *key, obs_copy_destination_object_info *object_info,
      obs_upload_part_info *copypart, obs_put_properties *put_properties,
      server_side_encryption_params *encryption_params,obs_response_handler *handler,
      void *callback_data);
```

请求参数说明

表 10-126 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
object_info	**obs_copy_destination_object_info** *	必选	**参数解释：** 指明多段上传任务。 **约束限制：** 无
copypart	**obs_upload_part_info** *	必选	**参数解释：** 上传段信息。 **约束限制：** 无
put_properties	**obs_put_properties***	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 服务端加密设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-127 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-128 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 10-129 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-130 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-131 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-132 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 10-133 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-134 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-135 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-136 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **取值范围：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 10-137 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 10-138 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-139 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties** *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-140 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 10-141 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-142 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-143 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-144 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 10-145 obs_copy_destination_object_info

参数名称	参数类型	是否必选	描述
destination_bucket	char *	必选	**参数解释：** 复制对象的目标桶。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
destination_key	char *	必选	**参数解释：** 复制对象的目标对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
version_id	char *	必选	**参数解释：** 复制对象的源对象版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
last_modified_return	int64_t *	必选	**参数解释：** 复制对象的最后修改时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
etag_return_size	int	必选	**参数解释：** 复制对象返回的etag长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
etag_return	char *	必选	**参数解释：** 复制对象返回的etag。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-146 obs_upload_part_info

参数名称	参数类型	是否必选	描述
part_number	unsigned int	必选	**参数解释:** 上传段的段号。 **约束限制:** 无 **取值范围:** 从1到10000的整数。 **默认取值:** 无
upload_id	int	必选	**参数解释:** 多段上传任务ID。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 10-147 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释:** 指定对象被下载时的文件类型。 **约束限制:** 无 **取值范围:** 参见HTTP标准头域Content-Type的取值。 **默认取值:** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	obs_canned_acl	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见obs_canned_acl。
az_redundancy	obs_az_redundancy	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见obs_az_redundancy。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 10-148 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 10-149 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 10-150 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 10-151 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为 application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-152 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见 image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-153 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

代码示例：复制段

以下示例展示如何通过copy_part进行分段复制：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何通过copy_part进行分段复制：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
```

```
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    char * destBucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    options.bucket_options.protocol = OBS_PROTOCOL_HTTP;
    // SSE-KMS加密相关参数，没有就不设置
    server_side_encryption_params encryption_params;
    memset(&encryption_params, 0, sizeof(server_side_encryption_params));
    // 复制段后返回的etag值
    char etag_return[256] = { 0 };
    char *key = "example_upload_file_object_key";
        // 定义复制段信息
    obs_copy_destination_object_info object_info;
    memset(&object_info, 0, sizeof(obs_copy_destination_object_info));
    object_info.destination_bucket = destBucketName;
    object_info.destination_key = "example_multi_part_upload_object_key";
    object_info.etag_return = etag_return;
    object_info.etag_return_size = 256;
    obs_upload_part_info copypart;
    memset(©part, 0, sizeof(obs_upload_part_info));
    // 设置响应回调函数
    obs_response_handler responseHandler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    char* upload_id = "000001930065F7C64014C38C138D654B";
    uint64_t object_size_total = 145184256;
    uint64_t part_size_one = 5 * 1024 * 1024;
    // 复制第一段
    copypart.part_number = 1;
    copypart.upload_id = upload_id;
    obs_status ret_status = OBS_STATUS_BUTT;
    //初始化put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 第一段复制前5mb
    put_properties.start_byte = 0;
    put_properties.byte_count = part_size_one;
    copy_part(&options, key, &object_info, ©part,
        &put_properties, &encryption_params, &responseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status && etag_return[0] != '\0') {
        printf(" copy part 1 successfully etag:%s. \n", etag_return);
    }
    else
    {
        printf("copy part 1 failed(%s).\n", obs_get_status_name(ret_status));
    }
    memset(etag_return, 0, sizeof(etag_return));
    // 复制第二段，复制剩下的数据
    copypart.part_number = 2;
    copypart.upload_id = upload_id;
    ret_status = OBS_STATUS_BUTT;
    put_properties.start_byte = part_size_one;
    put_properties.byte_count = object_size_total - part_size_one;
    copy_part(&options, key, &object_info, ©part,
        &put_properties, &encryption_params, &responseHandler, &ret_status);
    if (ret_status == OBS_STATUS_OK && etag_return[0] != '\0') {
        printf(" copy part 2 successfully etag:%s. \n", etag_return);
    }
    else
```

```
    {
        printf("copy part 2 failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
```



```
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于分段上传-复制段的API说明，请参见**拷贝段**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

10.8 分段上传-取消分段上传任务(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

通过分段上传任务的ID，取消指定桶中的分段上传任务。

您可以选择取消多段上传任务，取消多段上传任务之后无法再次使用该上传ID上传任何段。然后，OBS将释放被取消的多段上传任务中的每个段数据的所有存储。如果有多段上传已在进行中，即使您已执行中止操作，它们仍可以上传成功或失败。如果要释放所有分段使用的所有存储，必须在完成所有多段上传后再取消多段上传任务。

接口约束

- 您必须是桶拥有者或拥有取消分段上传任务的权限，才能取消分段上传任务。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:AbortMultipartUpload权限，如果使用桶策略则需授予AbortMultipartUpload权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void abort_multi_part_upload(const obs_options *options, char *key, const char *upload_id,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 10-154 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
upload_id	char *	必选	**参数解释：** 指明多段上传任务ID。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-155 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 10-156 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_list_type**。

表 10-157 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 10-158 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-159 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 10-160 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 10-161 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 10-162 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值：** 无
temp_auth_callback	void(***temp_auth_callback**) (char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-163 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-164 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 10-165 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-166 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 10-167 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-168 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-169 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 10-170 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：取消分段上传任务

以下示例展示如何取消分段上传任务：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 分段上传任务的对象名
    char *key = "example_multi_part_upload_object_key";
    // 分段上传任务id。
    char *upload_id = "00000194178F7ACE4014F60E75720850";
    // 设置响应回调函数
    obs_response_handler handler = {
        &response_properties_callback,
```



```
    &response_complete_callback
};
// 取消分段上传任务
obs_status ret_status;
abort_multi_part_upload(&options, key, upload_id, &handler, &ret_status);
if (OBS_STATUS_OK == ret_status)
{
    printf("test abort_multi_part_upload successfully.\n");
}
else
{
    printf("test abort_multi_part_upload faied(%s).\n", obs_get_status_name(ret_status));
}
// 释放分配的全局资源
obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
if (properties == NULL)
{
    printf("error! obs_response_properties is null!");
    if (callback_data != NULL)
    {
        obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
        printf("server_callback buf is %s, len is %llu",
            data->buffer, data->buffer_len);
        return OBS_STATUS_OK;
    }
    else {
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                              \
do {                                                    \
    if (properties-> field) {                               \
        printf("%s: %s\n", name, properties->field);        \
    }                                                   \
} while (0)
print_nonnull("request_id", request_id);
print_nonnull("request_id2", request_id2);
print_nonnull("content_type", content_type);
if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
}
print_nonnull("server", server);
print_nonnull("ETag", etag);
print_nonnull("expiration", expiration);
print_nonnull("website_redirect_location", website_redirect_location);
print_nonnull("version_id", version_id);
print_nonnull("allow_origin", allow_origin);
print_nonnull("allow_headers", allow_headers);
print_nonnull("max_age", max_age);
print_nonnull("allow_methods", allow_methods);
print_nonnull("expose_headers", expose_headers);
print_nonnull("storage_class", storage_class);
print_nonnull("server_side_encryption", server_side_encryption);
print_nonnull("kms_key_id", kms_key_id);
print_nonnull("customer_algorithm", customer_algorithm);
print_nonnull("customer_key_md5", customer_key_md5);
print_nonnull("bucket_location", bucket_location);
print_nonnull("obs_version", obs_version);
print_nonnull("restore", restore);
print_nonnull("obs_object_type", obs_object_type);
print_nonnull("obs_next_append_position", obs_next_append_position);
print_nonnull("obs_head_epid", obs_head_epid);
print_nonnull("reserved_indicator", reserved_indicator);
```



```
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于分段上传-取消分段上传任务的API说明，请参见**取消多段上传任务**。
- 更多关于分段上传的代码示例，请参见**Github示例**。
- 分段上传过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

11 多版本控制(C SDK)

11.1 设置桶的多版本状态(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持多版本控制，您可以在一个桶中保留对象的多个版本，使您更方便地检索和还原各个版本，在意外操作或应用程序故障时快速恢复数据。更多多版本相关信息请参见**多版本控制**。

调用设置桶的多版本状态接口，您可以为指定桶设置多版本状态。

接口约束

- 您必须是桶拥有者或拥有设置桶的多版本状态的权限，才能设置桶的多版本状态。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketVersioning权限，如果使用桶策略则需授予PutBucketVersioning权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_version_configuration(const obs_options *options, const char *version_status,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 11-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释:** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制:** 无
version_status	char *	必选	**参数解释:** 表示桶的多版本状态。 **约束限制:** 无 **取值范围:** • OBS_VERSION_STATUS_ENABLED • OBS_VERSION_STATUS_SUSPENDED **默认取值:** 无
handler	**obs_response_handler** *	必选	**参数解释:** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制:** 无
callback_data	void *	可选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 11-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_list_type**。

表 11-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-5 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-6 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-7 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-8 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-11 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-12 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-13 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-14 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-15 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-16 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-17 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置桶的多版本状态

以下示例展示如何设置桶的多版本状态：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置桶的多版本状态：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    // 设置桶的存储类别，此处以标准存储为例
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_STANDARD;
    obs_status ret_status = OBS_STATUS_BUTT;
    // 开启桶的多版本控制
    set_bucket_version_configuration(&options, OBS_VERSION_STATUS_ENABLED,
```



```
        &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket version successfully. \n");
    }
    else
    {
        printf("set bucket version failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
```



```
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶的多版本状态的API说明，请参见**设置桶的多版本状态**。
- 更多关于设置桶的多版本状态的代码示例，请参见**Github示例**。
- 设置桶的多版本状态过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 多版本控制相关问题请参见**多版本控制相关常见问题**。

11.2 获取桶的多版本状态(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持多版本控制，您可以在一个桶中保留对象的多个版本，使您更方便地检索和还原各个版本，在意外操作或应用程序故障时快速恢复数据。更多多版本相关信息请参见**多版本控制**。

调用获取桶的多版本状态接口，您可以获取指定桶的多版本状态。

接口约束

- 您必须是桶拥有者或拥有获取桶的多版本状态的权限，才能获取桶的多版本状态。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketVersioning权限，如果使用桶策略则需授予GetBucketVersioning权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_version_configuration(const obs_options *options, int status_return_size,
    char *status_return, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 11-18 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
status_return_size	int	必选	**参数解释：** 多版本状态缓存大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
status_return	char *	必选	**参数解释：** 多版本状态缓存。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-19 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-20 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 11-21 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-22 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-23 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-24 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-25 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-26 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-27 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-28 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-29 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-30 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-31 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-32 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-33 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-34 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶的多版本状态

以下示例展示如何获取桶的多版本状态：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何获取桶的多版本状态：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    // 设置桶的存储类别，此处以标准存储为例
    options.bucket_options.storage_class = OBS_STORAGE_CLASS_STANDARD;
    obs_status ret_status = OBS_STATUS_BUTT;
    // 获取桶的多版本状态
    char status[OBS_COMMON_LEN_256 + 1] = { 0 };
```

```
    get_bucket_version_configuration(&options, sizeof(status), status,
        &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("get bucket version successfully.\n policy=(%s)\n", status);
    }
    else
    {
        printf("get bucket version failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
```



```
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于获取桶的多版本状态的API说明，请参见**获取桶的多版本状态**。
- 更多关于获取桶的多版本状态的代码示例，请参见**Github示例**。
- 获取桶的多版本状态过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 多版本控制相关问题请参见**多版本控制相关常见问题**。

11.3 获取多版本对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

您可以通过get_object接口传入版本号（obs_object_info.version_id）来获取多版本对象。

如果版本号为空，则默认下载最新版本的对象。

接口约束

- 您必须是桶拥有者或拥有下载对象的权限，才能下载多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObject权限，如果使用桶策略则需授予GetObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 对于存储类别为归档存储或深度归档存储的多版本对象，需要确认多版本对象的状态为“已恢复”才能对其进行下载。

方法定义

```
void get_object(const obs_options *options, obs_object_info *object_info,
        obs_get_conditions *get_conditions,
        server_side_encryption_params *encryption_params,
        obs_get_object_handler *handler, void *callback_data);
```

请求参数

表 11-35 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
get_conditions	**obs_get_conditions** *	必选	**参数解释：** 对象筛选条件和读取范围设置。 **约束限制：** 无
encryption_params	**server_side_encryption_params** *	可选	**参数解释：** 获取对象的解密设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_get_object_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-36 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-37 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 11-38 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-39 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-40 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-41 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 11-42 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-43 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-44 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-45 obs_get_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
get_object_data_callback	obs_get_object_data_callback *	必选	**参数解释：** 回调函数指针，用于把要下载的数据从数据缓冲区复制到用户自定义回调数据。 **约束限制：** 无

表 11-46 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-47 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-48 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-49 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ== **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-50 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-51 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-52 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-53 obs_get_object_data_callback

参数名称	参数类型	是否必选	描述
buffer_size	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
buffer	char *	必选	**参数解释：** 待下载的数据缓冲区，把这个缓冲区的数据复制到callback_data中，实现下载数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-54 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-55 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 11-56 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-57 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 11-58 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。

参数名称	参数类型	是否必选	描述
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围**： ● kms ● AES256 **默认取值：** 无
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无

参数名称	参数类型	是否必选	描述
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 11-59 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

代码示例：获取多版本对象

以下示例展示如何get_object函数下载多版本对象到本地：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
```

```
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何通过get_object函数下载多版本对象到本地文件：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    obs_object_info object_info;
    // 设置下载的多版本对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = "G00111934EA2FAC2000040180A3CB360";
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 下载起始位置，默认0，从对象0字节位置开始下载
    getcondition.start_byte = 0;
    // 下载长度，默认0，一直下载到对象尾
    // getcondition.byte_count = 0;
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
```



```
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
```



```
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

11.4 复制多版本对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

用户可以根据需要将存储在OBS上的多版本对象复制到其他路径下。复制多版本对象操作将创建需要复制的多版本对象的副本，即为指定桶中的多版本对象创建一个副本。在单次操作中，您可以创建最大5GB的多版本对象副本。

接口约束

- 您必须是桶拥有者或拥有复制对象的权限，才能复制多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObject权限，如果使用桶策略则需授予PutObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 复制多版本对象操作的请求需要通过头域携带复制的源桶和多版本对象信息，不能携带消息实体。
- 目标多版本对象大小范围是[0, 5GB]，如果源多版本对象大小超过5GB，只能使用**分段上传-复制段(C SDK)**功能复制部分多版本对象。

方法定义

```
void copy_object(const obs_options *options, char *key, const char *version_id,
        obs_copy_destination_object_info *object_info,
        unsigned int is_copy, obs_put_properties *put_properties,
        server_side_encryption_params *encryption_params,
        obs_response_handler *handler, void *callback_data);
```

请求参数

表 11-60 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
object_info	obs_copy_destination_object_info *	必选	**参数解释：** 指明复制对象信息。 **约束限制：** 无
is_copy	unsigned int	必选	**参数解释：** 用来指定新对象的元数据是从源对象中复制，还是用请求中的元数据替换。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
put_properties	obs_put_properties*	可选	**参数解释：** 上传对象属性。 **约束限制：** 无
encryption_params	server_side_encryption_params *	可选	**参数解释：** 服务端加密设置。 **约束限制：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-61 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-62 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 11-63 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-64 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-65 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-66 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-67 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-68 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 11-69 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-70 server_side_encryption_params

参数名称	参数类型	是否必选	描述
encryption_type	obs_encryption_type	可选	**参数解释：** 加密类型。 **约束限制：** 无 **取值范围：** 请详见obs_encryption_type。
kms_server_side_encryption	char *	可选	**参数解释：** 使用该头域表示对象使用的服务端加密方式是SSE-KMS。 **约束限制：** 无 **取值范围：** • kms • AES256 **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了kms_server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示加密对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** AES256（指AES256加密算法） **默认取值：** 无
ssec_customer_key	char *	可选	**参数解释：** 该头域表示加密对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 256位密钥的base64编码。 **默认取值：** 无
des_ssec_customer_algorithm	char *	可选	**参数解释：** 该头域表示解密源对象使用的算法。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
des_ssec_customer_key	char *	可选	**参数解释：** 该头域表示解密源对象使用的密钥。 **约束限制：** 仅SSE-C方式下使用该头域。 **取值范围：** 无 **默认取值：** 无

表 11-71 obs_encryption_type

枚举值	说明
OBS_ENCRYPTION_KMS	使用KMS加密方式。
OBS_ENCRYPTION_SSEC	使用SSE-C加密方式。

表 11-72 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-73 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-74 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-75 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-76 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-77 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-78 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-79 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
meta_data	obs_name_value*	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	metadata_action_indicator	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见metadata_action_indicator。
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 11-80 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 11-81 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）

参数名称	参数类型	是否必选	描述
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 11-82 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 11-83 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为 application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-84 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-85 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。
obs_image_process_style	图片处理参数以style开头。

表 11-86 obs_copy_destination_object_info

参数名称	参数类型	是否必选	描述
destination_bucket	constchar*	必选	**参数解释：** 复制的目标桶名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
destination_key	constchar*	必选	**参数解释：** 复制的目标对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	constchar*	可选	**参数解释：** 目标对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
last_modified_return	int64_t *	必选	**参数解释：** 对象最近一次被修改的时间（UTC时间）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
etag_return_size	int	必选	**参数解释：** etag_return的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
etag_return	char*	必选	**参数解释：** 新对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无

您可以通过copy_object接口传入版本号（version_id）来复制多版本对象，代码示例如下：

代码示例：复制多版本对象

以下示例展示如何通过copy_object函数复制多版本对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过copy_object函数复制多版本对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    char eTag[OBS_COMMON_LEN_256] = { 0 };
    int64_t lastModified;
    // 设置目的对象信息
    obs_copy_destination_object_info objectinfo = { 0 };
    objectinfo.destination_bucket = "example-dest-bucket-name";
    objectinfo.destination_key = "example_destination_key";
    objectinfo.etag_return = eTag;
```

```
    objectinfo.etag_return_size = sizeof(eTag);
    objectinfo.last_modified_return = &lastModified;
    // 设置响应回调函数
    obs_response_handler responseHandler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* source_object_key = "example_source_object_key";
    char* source_object_version_id = "G00111934EAA2CE900004016129AC990";
    // 复制对象
    copy_object(&options, source_object_key, source_object_version_id, &objectinfo, 1, NULL, NULL,
&responseHandler, &ret_status);
    if (OBS_STATUS_OK == ret_status && eTag[0] != '\0') {
        printf("test_copy_object successfully. \n");
    }
    else
    {
        printf("test_copy_object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
```



```
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

11.5 恢复多版本归档存储对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

如果要获取归档或深度归档存储多版本对象的内容，需要先将多版本对象恢复，然后再执行下载数据的操作。多版本对象恢复后，会产生一个标准存储类别的多版本对象副本，也就是说会同时存在标准存储类别的多版本对象副本和归档或深度归档存储多版本对象，在恢复多版本对象的保存时间到期后标准存储类别的多版本对象副本会自动删除。

该接口可以恢复指定桶中的归档或深度归档存储多版本对象。

接口约束

- 您必须是桶拥有者或拥有恢复归档或深度归档存储对象的权限，才能恢复归档或深度归档存储多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:RestoreObject权限，如果使用桶策略则需授予RestoreObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 重复恢复归档或深度归档存储多版本对象时在延长恢复有效期的同时，也将会对恢复时产生的恢复费用进行重复收取。产生的标准存储类别的多版本对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

方法定义

```
void restore_object(const obs_options *options, obs_object_info *object_info, const char *days,
      obs_tier tier,const obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 11-87 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。

参数名称	参数类型	是否必选	描述
days	char *	必选	**参数解释：** 指定恢复对象的保存时间。 **约束限制：** 无 **取值范围：** [1，30]，单位：天。 **默认取值：** 无
tier	**obs_tier**	可选	**参数解释：** 恢复选项，可以为：obs_tier.OBS_TIER_EXPEDITED，obs_tier.OBS_TIER_STANDARD。 **约束限制：** 无 **取值范围：** 请详见**obs_tier**。
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-88 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-89 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 11-90 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-91 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-92 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-93 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-94 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-95 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-96 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-97 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** ● 同一个桶中存储的对象名必须是唯一的。 ● 指定的对象必须是归档存储类别，否则调用该接口会报错。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-98 obs_tier

枚举值	说明
OBS_TIER_NULL	默认的无效值
OBS_TIER_STANDARD	表示标准恢复对象，归档存储恢复耗时3~5 h，深度归档（受限公测）存储恢复约耗时5~12 h。
OBS_TIER_EXPEDITED	表示快速恢复对象，归档存储恢复耗时1~5 min，深度归档（受限公测）存储恢复约耗时3~5 h。

表 11-99 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-100 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-101 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-102 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 密钥ID获取方法请参见**查看密钥**。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-103 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-104 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-105 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

您可以通过restore_object接口传入版本号（obs_object_info.version_id）来恢复多版本归档存储对象，代码示例如下：

代码示例：恢复多版本归档存储对象

以下示例展示如何通过restore_object接口传入版本号（obs_object_info.version_id）来恢复多版本归档存储对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何通过restore_object接口传入版本号（obs_object_info.version_id）来恢复多版本归档存储对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
```



```
    options.bucket_options.bucket_name = bucketName;
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test_restore";
    object_info.version_id = "G00111934ED39BD60000401401EDE57B";
    // 设置响应回调函数
    obs_response_handler handler =
    {
      &response_properties_callback, &response_complete_callback
    };
    // 恢复对象
    obs_tier tier = OBS_TIER_EXPEDITED;
    obs_status ret_status = OBS_STATUS_BUTT;
    const char* storedDays = "1";
    options.request_options.auth_switch = OBS_OBS_TYPE;
    restore_object(&options, &object_info, storedDays, tier, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("restore object successfully. \n");
    }
    else
    {
        printf("restore object faied(%s).\n", obs_get_status_name(ret_status));
        return;
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *data =
            (obs_status *)callback_data;
        *data = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
```



```
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
```


11.6 列举桶内多版本对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

用列举桶内多版本对象接口，可列举指定桶内的部分或所有多版本对象的描述信息。您还可以通过设置前缀、数量、起始位置等参数，返回符合您筛选条件的多版本对象信息。返回结果以多版本对象名的字典序排序。

接口约束

- 您必须是桶拥有者或拥有列举桶内多版本对象的权限，才能列举桶内多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:ListBucketVersions权限，如果使用桶策略则需授予ListBucketVersions权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void list_versions(const obs_options *options, const char *prefix, const char *key_marker,
        const char *delimiter, int maxkeys, const char *version_id_marker,
        obs_list_versions_handler *handler, void *callback_data);
```

请求参数说明

表 11-106 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
prefix	char *	可选	**参数解释：** 限定返回的对象名必须带有prefix前缀。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
key_marker	char *	可选	**参数解释：** 列举多版本对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
delimiter	char *	可选	**参数解释：** 用于对对象名进行分组的字符。对于对象名中包含delimiter的对象，其对象名（如果请求中指定了prefix，则此处的对象名需要去掉prefix）中从首字符至第一个delimiter之间的字符串将作为一个分组并作为common_prefixes返回。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
version_id_marker	char *	可选	**参数解释：** 与key_marker配合使用，返回的对象列表将是对象名和版本号按照字典序排序后该参数以后的所有对象。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
maxkeys	int	必选	**参数解释：** 列举对象的最大数目。 **约束限制：** 当该参数超出1000时，按照默认的1000进行处理。 **取值范围：** [1，1000]，单位：个。 **默认取值：** 1000
handler	**obs_list_versions_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-107 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-108 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_list_type**。

表 11-109 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-110 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-111 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-112 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-113 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-114 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-115 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-116 obs_list_versions_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
list_versions_callback	**obs_list_versions_callback** *	必选	**参数解释：** 标签回调函数指针，可以在这个回调中把列举结果的具体内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-117 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-118 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-119 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-120 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000

参数名称	参数类型	是否必选	描述
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-121 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	**obs_name_value***	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
error_headers	char**	**参数解释:** 响应头域中包含error的所有头域。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 11-122 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
value	char *	可选	**参数解释:** 属性的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 11-123 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。

枚举值	说明
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-124 obs_list_versions_callback

参数名称	参数类型	是否必选	描述
is_truncated	int	必选	**参数解释：** 返回的结果列表是否截断。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
next_key_marker	const char *	必选	**参数解释：** 下次列举多版本对象请求的起始位置。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的key_marker值。 **约束限制：** 该字段仅用于多版本列举。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
next_versionid_marker	const char *	必选	**参数解释**： 下次列举多版本对象请求的起始位置，与next_key_marker配合使用。如果本次没有返回全部结果，响应请求中将包含该元素，用于标明接下来请求的version_id_marker值。 **约束限制：** 该字段只适用于多版本列举场景。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
versions	const **obs_list_versions***	必选	**参数解释：** 列举出的多版本相关信息。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-125 obs_list_versions

参数名称	参数类型	是否必选	描述
bucket_name	const char *	可选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
prefix	const char *	可选	**参数解释：** 限定返回的对象名必须带有prefix前缀。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
key_marker	const char *	可选	**参数解释：** 列举多版本对象的起始位置，返回的对象列表将是对象名按照字典序排序后该参数以后的所有对象。 **约束限制：** 该字段仅用于多版本列举。 **取值范围：** 上次请求返回体的nextKeyMarker值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
delimiter	const char *	可选	**参数解释：** 将对象名进行分组的分隔符。如果指定了prefix，从prefix到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条CommonPrefixes；如果没有指定prefix，从对象名的首字符到第一次出现delimiter间具有相同字符串的对象名会被分成一组，形成一条CommonPrefixes。 例如，桶中有3个对象，分别为abcd、abcde、bbcde。如果指定delimiter为d，prefix为a，abcd、abcde会被分成一组，形成一条前缀为abcd的commonPrefix；如果只指定delimiter为d，abcd、abcde会被分成一组，形成一条前缀为abcd的commonPrefix，而bbcde会被单独分成一组，形成一条前缀为bbcd的commonPrefix。 对于并行文件系统，不携带此参数时默认列举是递归列举此目录下所有内容，会列举子目录。在大数据场景下（目录层级深、目录下文件多）的列举，建议设置[delimiter=/]，只列举当前目录下的内容，不列举子目录，提高列举效率。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
max_keys	const char *	可选	**参数解释：** 列举多版本对象的最大数目，返回的对象列表将是按照字典顺序的最多前maxKeys个对象。 **约束限制：** 当该参数超出1000时，按照默认的1000进行处理。 **取值范围：** [1，1000]，单位：个。 **默认取值：** 1000
versions	**obs_version** *	可选	**参数解释：** 对象信息 **约束限制：** 无

参数名称	参数类型	是否必选	描述
versions_count	int	可选	**参数解释：** versions的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
common_prefixes	const char **	可选	**参数解释：** 当请求中设置了delimiter分组字符时，返回按delimiter分组后的对象名称前缀列表。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
common_prefixes_count	int	可选	**参数解释：** common_prefixes的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-126 obs_version

参数名称	参数类型	是否必选	描述
key	const char *	可选	**参数解释：** 对象名。对象名是对象在存储桶中的唯一标识。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
is_latest	const char *	可选	**参数解释：** 标识对象是否是最新的版本。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
last_modified	int64_t	可选	**参数解释：** 对象最近一次被修改的时间（UTC时间）。 **约束限制：** 无 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释**： 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。实际的ETag是对象的哈希值。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，通过MD5加密后都有唯一的ETag。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
size	uint64_t	可选	**参数解释：** 对象的字节数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
owner_id	const char *	可选	**参数解释：** 对象拥有者的domain_id（账号ID）。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
is_delete	const char *	可选	**参数解释：** 是否是删除标记。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：列举多版本对象

以下示例展示如何通过函数list_versions列举出桶里的多版本对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
typedef struct list_versions_callback_data
{
    char bucket_name[1024];
    char prefix[1024];
    char key_marker[1024];
    char delimiter[1024];
    int max_keys;
    int is_truncated;
    char next_key_marker[1024];
    char next_versionId_marker[1024];
    int keyCount;
    int allDetails;
    obs_status ret_status;
} list_versions_callback_data;
static void list_versions_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
static obs_status listVersionsCallback(int is_truncated, const char *next_key_marker, const char
*next_versionId_marker,
    const obs_list_versions *list_versions, void *callback_data);
int main()
{
    // 以下示例展示如何通过函数list_versions列举出桶里的多版本对象。
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
```

```
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    char * prefix = "example";
    char * key_marker = "";
    char * delimiter = "/";
    int maxkeys = 1000;
    // 设置响应回调函数
    obs_list_versions_handler list_versions_handler =
    {
        { &response_properties_callback, &list_versions_complete_callback },
        &listVersionsCallback
    };
    // 创建并初始化回调数据
    list_versions_callback_data data;
    char* version_id_marker = NULL;
    memset(&data, 0, sizeof(list_versions_callback_data));
    data.ret_status = OBS_STATUS_BUTT;
    snprintf(data.next_key_marker, sizeof(data.next_key_marker), "%s", key_marker);
    if (version_id_marker)
    {
        snprintf(data.next_versionId_marker, sizeof(data.next_versionId_marker), "%s", version_id_marker);
    }
    data.keyCount = 0;
    data.allDetails = 1;
    data.is_truncated = 0;
    // 列举多版本对象，通过prefix指定对象前缀，maxkeys指定数目列举，可实现分页列举，
    // delimiter指定分组列举的字符，按文件夹分组，则delimiter设置为“/”，
    // 通过key_marker指定多版本起始位置，version_id_marker指定多版本起始版本号
    list_versions(&options, prefix, key_marker, delimiter, maxkeys, version_id_marker,
        &list_versions_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("list versions successfully. \n");
    }
    else
    {
        printf("list versions failed(%s).\n", obs_get_status_name(data.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
```



```
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
static void list_versions_complete_callback(obs_status status,
```

```
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data) {
        list_versions_callback_data *data = (list_versions_callback_data*)callback_data;
        data->ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
static void printListVersionsHeader(int allDetails)
{
    printf("%-30s  %-20s  %-5s  %-32s",
        "              Key",
        "    Last Modified", " Size", "           VersionId");
    if (allDetails) {
        printf("  %-34s  %-12s  %-12s  %-12s",
            "               ETag",
            "Storage Class",
            "Owner ID",
            "                    Display Name");
    }
    printf("\n");
    printf("------------------------------  "
        "--------------------  -----  --------------------------------");
    if (allDetails) {
        printf("  ----------------------------------  ------------  --------------------------------   ------------");
    }
    printf("\n");
}
static obs_status listVersionsCallback(int is_truncated, const char *next_key_marker, const char
*next_versionId_marker,
    const obs_list_versions *list_versions, void *callback_data)
{
    list_versions_callback_data *data =
        (list_versions_callback_data *)callback_data;
    data->is_truncated = is_truncated;
    if ((!next_key_marker || !next_key_marker[0]) && list_versions->versions_count) {
        next_key_marker = list_versions->versions[list_versions->versions_count - 1].key;
    }
    if (next_key_marker) {
        snprintf(data->next_key_marker, sizeof(data->next_key_marker), "%s", next_key_marker);
    }
    else {
        data->next_key_marker[0] = 0;
    }
    if ((!next_versionId_marker || !next_versionId_marker[0]) && list_versions->versions_count) {
        next_versionId_marker = list_versions->versions[list_versions->versions_count - 1].version_id;
    }
    if (next_versionId_marker) {
        snprintf(data->next_versionId_marker, sizeof(data->next_versionId_marker), "%s",
next_versionId_marker);
    }
    else {
        data->next_versionId_marker[0] = 0;
    }
    if (NULL != list_versions->bucket_name)
    {
        printf("Name = %s\n", list_versions->bucket_name);
    }
    if (NULL != list_versions->prefix)
    {
        printf("prefix = %s\n", list_versions->prefix);
    }
    if (NULL != list_versions->key_marker)
    {
        printf("key_marker = %s\n", list_versions->key_marker);
```



```
    }
    if (NULL != list_versions->delimiter)
    {
        printf("delimiter = %s\n", list_versions->delimiter);
    }
    if (NULL != list_versions->max_keys)
    {
        printf("max_keys = %s\n", list_versions->max_keys);
    }
    if (list_versions->versions_count && !data->keyCount) {
        printListVersionsHeader(data->allDetails);
    }
    int i;
    for (i = 0; i < list_versions->versions_count; i++) {
        obs_version *version = &(list_versions->versions[i]);
        char timebuf[256] = { 0 };
        time_t t = (time_t)version->last_modified;
        strftime(timebuf, sizeof(timebuf), "%Y-%m-%dT%H:%M:%SZ",
            gmtime(&t));
        char sizebuf[16] = { 0 };
        if (version->size < 100000) {
            sprintf_s(sizebuf, sizeof(sizebuf), "%5llu", (unsigned long long) version->size);
        }
        else if (version->size < (1024 * 1024)) {
            sprintf_s(sizebuf, sizeof(sizebuf), "%4lluK",
                ((unsigned long long) version->size) / 1024ULL);
        }
        else if (version->size < (10 * 1024 * 1024)) {
            float f = version->size;
            f /= (1024 * 1024);
            sprintf_s(sizebuf, sizeof(sizebuf), "%1.2fM", f);
        }
        else if (version->size < (1024 * 1024 * 1024)) {
            sprintf_s(sizebuf, sizeof(sizebuf), "%4lluM",
                ((unsigned long long) version->size) /
                (1024ULL * 1024ULL));
        }
        else {
            float f = (version->size / 1024);
            f /= (1024 * 1024);
            sprintf_s(sizebuf, sizeof(sizebuf), "%1.2fG", f);
        }
        printf("%-30s  %s  %s  %-32s", version->key, timebuf, sizebuf, version->version_id);
        if (data->allDetails) {
            printf("  %-34s  %-12s  %-32s  %-12s",
                version->etag,
                version->storage_class,
                version->owner_id ? version->owner_id : "",
                version->owner_display_name ?
                version->owner_display_name : "");
        }
        printf("\n");
    }
    printf("versions_count=%d\n", list_versions->versions_count);
    printf("--------------------------------------------------------------------------------");
    printf("--------------------------------");
    printf("--------------------------------------------------------------------------------\n");
    for (i = 0; i < list_versions->common_prefixes_count; i++)
    {
        printf("commonPrefix => prefix = %s\n", *(list_versions->common_prefixes + i));
    }
    data->keyCount += list_versions->versions_count;
    return OBS_STATUS_OK;
}
```

说明

- 每次至多返回1000个多版本对象，如果指定桶包含的对象数量大于1000，则返回结果中is_truncated为true表明本次没有返回全部对象，并可通过next_key_marker和next_versionId_marker获取下次列举的起始位置。
- 如果想获取指定桶包含的所有多版本对象，可以采用分页列举的方式。

相关链接

- 关于列举桶内多版本对象的API说明，请参见**列举桶内对象**。
- 更多关于列举桶内多版本对象的代码示例，请参见**Github示例**。
- 列举桶内多版本对象过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 桶和对象相关常见问题请参见**桶和对象相关常见问题**。

11.7 设置多版本对象 ACL-为对象设置预定义访问策略(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS支持对多版本对象的操作进行权限控制。默认情况下，只有多版本对象的创建者才有该多版本对象的读写权限。用户也可以设置其他的访问策略，比如对一个多版本对象可以设置公共访问策略，允许所有人对其都有读权限。SSE-KMS方式加密的多版本对象即使设置了ACL，跨租户也不生效。

OBS用户在上传多版本对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的多版本对象更改或者获取ACL(access control list) 。

可以通过本接口设置指定桶中多版本对象的访问权限。

接口约束

- 您必须是桶拥有者或拥有设置对象ACL的权限，才能设置多版本对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObjectAcl权限，如果使用桶策略则需授予PutObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个多版本对象的ACL最多支持配置100条授权策略。

方法定义

```
void set_object_acl_by_head(const obs_options *options, obs_object_info *object_info,
    obs_canned_acl canned_acl, obs_response_handler *handler, void *callback_data);
```


请求参数说明

表 11-127 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
canned_acl	**obs_canned_acl**	必选	**参数解释：** 预定义访问策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-128 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-129 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 11-130 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-131 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-132 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-133 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-134 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-135 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 11-136 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-137 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-138 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-139 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-140 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-141 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-142 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-143 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-144 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-145 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

代码示例：设置多版本对象 ACL

以下示例展示如何通过set_object_acl_by_head来设置多版本对象访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过set_object_acl_by_head来设置对象属性：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
```



```
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    obs_canned_acl canned_acl = OBS_CANNED_ACL_PUBLIC_READ_WRITE;
    obs_object_info object_info;
    object_info.key = "example_get_file_test";
    object_info.version_id = "G00111934EAA2CE900004016129AC990";
    // 设置对象预定义访问策略
    set_object_acl_by_head(&options, &object_info, canned_acl, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set object acl by head successfully. \n");
    }
    else
    {
        printf("set object acl by head failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
```



```
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```


11.8 设置多版本对象 ACL-直接设置对象 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

OBS支持对多版本对象的操作进行权限控制。默认情况下，只有多版本对象的创建者才有该多版本对象的读写权限。用户也可以设置其他的访问策略，比如对一个多版本对象可以设置公共访问策略，允许所有人对其都有读权限。SSE-KMS方式加密的多版本对象即使设置了ACL，跨租户也不生效。

OBS用户在上传多版本对象时可以设置权限控制策略，也可以通过ACL操作API接口对已存在的多版本对象更改或者获取ACL(access control list) 。

可以通过本接口设置指定桶中多版本对象的访问权限。

接口约束

- 您必须是桶拥有者或拥有设置对象ACL的权限，才能设置多版本对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:PutObjectAcl权限，如果使用桶策略则需授予PutObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 一个多版本对象的ACL最多支持配置100条授权策略。

方法定义

```
void set_object_acl(const obs_options *options, manager_acl_info *aclinfo, obs_response_handler *handler,
void *callback_data);
```

请求参数说明

表 11-146 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无

参数名称	参数类型	是否必选	描述
aclinfo	manager_acl_info *	必选	**参数解释：** 管理ACL权限信息相关结构体。 **约束限制：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-147 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-148 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 11-149 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-150 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-151 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-152 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 11-153 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-154 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-155 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-156 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-157 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-158 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-159 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-160 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-161 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-162 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-163 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 如果非多版本对象，请把version设置为NULL
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。 **默认取值：** OBJECT_DELIVERED_TRUE（指对象ACL继承桶的ACL）
acl_grants	**obs_acl_grant** *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 11-164 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-165 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 11-166 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	obs_grantee_type	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见obs_grantee_type。

参数名称	参数类型	是否必选	描述
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 11-167 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 11-168 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 11-169 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

代码示例：设置多版本对象访问权限

以下示例展示如何通过set_object_acl来设置多版本对象访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过set_object_acl来设置多版本对象访问权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
```



```
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 定义多版本对象访问权限信息
    manager_acl_info aclinfo;
    memset(&aclinfo, 0, sizeof(aclinfo));
    aclinfo.object_info.key = "example_source_object_key";
    aclinfo.object_info.version_id = "G00111934EAA2CE900004016129AC990";
    aclinfo.owner_id = "1****************************a";
    // 设置acl
    int acl_grant_count = 1;
    aclinfo.acl_grant_count_return = &acl_grant_count;
    aclinfo.acl_grants = (obs_acl_grant*)malloc(sizeof(obs_acl_grant) * acl_grant_count);
    // 设置被授权账号的租户ID(canonical_user.id)
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.id, "0****************************0");
    strcpy(aclinfo.acl_grants[0].grantee.canonical_user.display_name, "name1");
    aclinfo.acl_grants[0].grantee_type = OBS_GRANTEE_TYPE_CANONICAL_USER;
    // 设置ACL权限，此处以读权限为例
    aclinfo.acl_grants[0].permission = OBS_PERMISSION_READ;
    // 设置对象访问权限
    set_object_acl(&options, &aclinfo, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set object acl successfully. \n");
    }
    else
    {
        printf("set object acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    free(aclinfo.acl_grants);
    aclinfo.acl_grants = NULL;
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
```

```
    do {                                                \
        if (properties-> field) {                       \
            printf("%s: %s\n", name, properties->field);        \
        }                                               \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
```



```
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

11.9 获取多版本对象 ACL(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持对桶操作进行权限控制，您可以为桶设置访问策略，指定某一个用户对某一个桶是否有权行使某一项指定操作。OBS权限控制的方式有IAM、桶策略和ACL三种，ACL按照粒度又分为桶ACL和对象ACL，本节将对多版本对象ACL接口进行详细介绍，更多权限相关内容可参见《 对象存储服务权限配置指南 》的**OBS权限控制概述**章节。

多版本对象ACL是跨账号场景的权限，设置授权的对象不是当前账号，也不是当前账号下的IAM用户，而是另一个华为云账号及其账号下的IAM用户；授权的范围是以对象为粒度的，一条ACL策略为一个对象设置策略，因此设置ACL策略时您必须明确指定对象名；多版本对象ACL授予的权限包括对象的访问权限和多版本对象ACL的访问权限两个方面，对象的访问权限包括对桶内对象的查看和编辑权限，多版本对象ACL的访问权限包括对多版本对象ACL策略的查看和编辑权限，详情可参见**ACL权限控制方式介绍**。

调用获取多版本对象ACL接口，您可以获取指定多版本对象的ACL策略。

接口约束

- 您必须是桶拥有者或拥有获取对象ACL的权限，才能获取多版本对象ACL。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:GetObjectAcl权限，如果使用桶策略则需授予GetObjectAcl权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的region以及region与endPoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_object_acl(const obs_options *options, manager_acl_info *aclinfo, obs_response_handler *handler,
void *callback_data);
```

请求参数说明

表 11-170 请求参数列表

参数名称	参数类型	是否必选	描述
options	const**obs_options** *	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权 **约束限制：** 无
aclinfo	**manager_acl_info** *	必选	**参数解释：** 管理ACL权限信息相关结构体。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-171 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-172 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）

参数名称	参数类型	是否必选	描述
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
storage_class	**obs_storage_class**	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见**obs_storage_class**。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 11-173 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。

枚举值	说明
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-174 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-175 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-176 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 11-177 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-178 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-179 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-180 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-181 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-182 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-183 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-184 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-185 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-186 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-187 manager_acl_info

参数名称	参数类型	是否必选	描述
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 如果非多版本对象，请把version设置为NULL
owner_id	char *	必选	**参数解释：** 桶所有者的账号ID，即domain_id。 **约束限制：** 无 **取值范围：** 如何获取桶所有者的账号ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无

参数名称	参数类型	是否必选	描述
owner_display_name	char *	必选	**参数解释：** 用户的显示名称。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grant_count_return	int *	必选	**参数解释：** acl_grants数组的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
object_delivered	**obs_object_delivered**	必选	**参数解释：** 对象ACL是否继承桶的ACL。 **约束限制：** 无 **取值范围：** 请详见**obs_object_delivered**。 **默认取值：** OBJECT_DELIVERED_TRUE（指对象ACL继承桶的ACL）
acl_grants	**obs_acl_grant** *	必选	**参数解释：** 权限信息结构体。 **约束限制：** 无

表 11-188 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-189 obs_object_delivered

枚举值	说明
OBJECT_DELIVERED_TRUE	对象ACL继承桶的ACL。
OBJECT_DELIVERED_FALSE	对象ACL不继承桶的ACL。

表 11-190 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	obs_grantee_type	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见obs_grantee_type。

参数名称	参数类型	是否必选	描述
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 11-191 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 11-192 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 11-193 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

代码示例：获取多版本对象访问权限

以下示例展示如何通过get_object_acl获取多版本对象的访问权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
manager_acl_info* malloc_acl_info();
void free_acl_info(manager_acl_info **acl);
void print_acl(manager_acl_info* acl);
int main()
{
    // 以下示例展示如何通过get_object_acl获取多版本对象的访问权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
```



```
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    options.bucket_options.protocol = OBS_PROTOCOL_HTTP;
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 对象访问权限信息
    manager_acl_info *aclinfo = malloc_acl_info();
    aclinfo->object_info.key = "example_source_object_key";
    aclinfo->object_info.version_id = "G00111934EAA2CE900004016129AC990";
    // 获取对象权限信息
    get_object_acl(&options, aclinfo, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("get object acl successfully. \n");
        print_acl(aclinfo);
    }
    else
    {
        printf("get object acl failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 销毁内存
    free_acl_info(&aclinfo);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
```



```
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    (void)callback_data;
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
```



```
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
void print_acl(manager_acl_info* acl) {
    if (acl == NULL) {
        printf("acl is NULL.\n");
        return;
    }
    printf("object key :%s\n", acl->object_info.key);
    printf("object version_id :%s\n", acl->object_info.version_id);
    printf("object owner_id :%s\n", acl->owner_id);
    printf("object owner_display_name :%s\n", acl->owner_display_name);
    switch (acl->object_delivered)
    {
    case OBJECT_DELIVERED_TRUE:
        printf("object acl delivered.\n");
        break;
    default:
        printf("object acl not delivered.\n");
        break;
    }
    if (acl->acl_grant_count_return == NULL) {
        printf("acl_grant_count_return is NULL.\n");
        return;
    }
    printf("object acl_grant_count :%d\n", *acl->acl_grant_count_return);
    if (acl->acl_grants == NULL) {
        printf("acl_grants is NULL.\n");
        return;
    }
    for (int i = 0; i < *acl->acl_grant_count_return; ++i) {
        obs_acl_grant* acl_grant = acl->acl_grants + i;
        printf("acl %d\n", i + 1);
        switch (acl_grant->bucket_delivered)
        {
        case BUCKET_DELIVERED_TRUE:
            printf("object acl delivered from bucket.\n");
            break;
        default:
            printf("object acl not delivered from bucket.\n");
            break;
        }
        switch (acl_grant->grantee_type)
        {
        case OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL.\n");
            break;
        case OBS_GRANTEE_TYPE_CANONICAL_USER:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_CANONICAL_USER.\n");
            break;
        case OBS_GRANTEE_TYPE_ALL_OBS_USERS:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_ALL_OBS_USERS.\n");
            break;
        case OBS_GRANTEE_TYPE_ALL_USERS:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_ALL_USERS.\n");
            break;
        case OBS_GRANTEE_TYPE_LOG_DELIVERY:
            printf("object acl grantee_type is OBS_GRANTEE_TYPE_LOG_DELIVERY.\n");
            break;
        default:
            printf("object acl grantee_type is unknown.\n");
            break;
        }
        switch (acl_grant->permission)
        {
        case OBS_PERMISSION_READ:
            printf("object acl grantee_type is OBS_PERMISSION_READ.\n");
            break;
```



```
        case OBS_PERMISSION_WRITE:
            printf("object acl grantee_type is OBS_PERMISSION_WRITE.\n");
            break;
        case OBS_PERMISSION_READ_ACP:
            printf("object acl grantee_type is OBS_PERMISSION_READ_ACP.\n");
            break;
        case OBS_PERMISSION_WRITE_ACP:
            printf("object acl grantee_type is OBS_PERMISSION_WRITE_ACP.\n");
            break;
        case OBS_PERMISSION_FULL_CONTROL:
            printf("object acl grantee_type is OBS_PERMISSION_FULL_CONTROL.\n");
            break;
        default:
            printf("object acl grantee_type is unknown.\n");
            break;
        }
        printf("acl_grant->grantee.canonical_user.display_name: %s\n", acl_grant-
>grantee.canonical_user.display_name);
        printf("acl_grant->grantee.canonical_user.id: %s\n", acl_grant->grantee.canonical_user.id);
    }
}
manager_acl_info* malloc_acl_info()
{
    manager_acl_info *aclinfo = (manager_acl_info*)malloc(sizeof(manager_acl_info));
    memset(aclinfo, 0, sizeof(manager_acl_info));
    size_t acl_grants_size = OBS_MAX_ACL_GRANT_COUNT * sizeof(obs_acl_grant);
    aclinfo->acl_grants = (obs_acl_grant*)malloc(acl_grants_size);
    memset(aclinfo->acl_grants, 0, acl_grants_size);
    aclinfo->acl_grant_count_return = (int*)malloc(sizeof(int));
    *(aclinfo->acl_grant_count_return) = 0;
    size_t owner_id_size = (OBS_MAX_GRANTEE_USER_ID_SIZE + 1) * sizeof(char);
    size_t owner_display_name_size = (OBS_MAX_GRANTEE_USER_ID_SIZE + 1) * sizeof(char);
    aclinfo->owner_id = (char *)malloc(owner_id_size);
    memset(aclinfo->owner_id, 0, owner_id_size);
    aclinfo->owner_display_name = (char *)malloc(owner_display_name_size);
    memset(aclinfo->owner_display_name, 0, owner_display_name_size);
    return aclinfo;
}
void free_acl_info(manager_acl_info **acl)
{
    manager_acl_info *aclinfo = *acl;
    free(aclinfo->acl_grants);
    free(aclinfo->owner_display_name);
    free(aclinfo->owner_id);
    free(aclinfo->acl_grant_count_return);
    free(aclinfo);
}
```

11.10 删除多版本对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

为节省空间和成本，您可以根据需要删除指定桶中的单个多版本对象。

接口约束

- 您必须是桶拥有者或拥有删除对象的权限，才能删除多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:DeleteObject权限，如果使用桶策略则需授予DeleteObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 桶没有开启多版本控制功能时，已删除的多版本对象不可恢复，请谨慎操作。

方法定义

```
void delete_object(const obs_options *options, obs_object_info *object_info,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 11-194 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-195 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure** *	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-196 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 11-197 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-198 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-199 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-200 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 11-201 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-202 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-203 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-204 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-205 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-206 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-207 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-208 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-209 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-210 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-211 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例

以下示例展示如何通过delete_object删除单个多版本对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
int main()
{
    // 以下示例展示如何通过delete_object删除单个多版本对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 要删除的对象信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_source_object_key";
    object_info.version_id = "G00111934EAA2CE900004016129AC990";
    // 设置响应回调函数
    obs_response_handler response_handler =
    {
        &response_properties_callback, &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 删除对象
    delete_object(&options, &object_info, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        printf("delete object successfully. \n");
    }
    else
    {
        printf("delete object failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
```



```
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
```



```
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

11.11 批量删除多版本对象(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

为节省空间和成本，您可以根据需要批量删除指定桶中的单个多版本对象。

接口约束

- 您必须是桶拥有者或拥有删除对象的权限，才能批量删除多版本对象。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:object:DeleteObject权限，如果使用桶策略则需授予DeleteObject权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 桶没有开启多版本控制功能时，已删除的多版本对象不可恢复，请谨慎操作。

方法定义

```
void batch_delete_objects(const obs_options *options, obs_object_info *object_info,obs_delete_object_info
*delobj,
                          obs_put_properties *put_properties, obs_delete_object_handler *handler, void
*callback_data);
```

请求参数说明

表 11-212 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
object_info	**obs_object_info** *	必选	**参数解释：** 对象名和版本号。 **约束限制：** 非多版本对象，version设置为0。
delobj	**obs_delete_object_info***	必选	**参数解释：** 删除对象个数和指定使用quiet模式。 **约束限制：** 无
put_properties	**obs_put_properties***	可选	**参数解释：** 设置删除对象的校验属性。 **约束限制：** 无
handler	**obs_delete_object_handler***	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-213 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 11-214 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 11-215 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 11-216 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-217 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 11-218 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 11-219 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 11-220 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-221 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-222 obs_delete_object_info

参数名称	参数类型	是否必选	描述
keys_number	unsigned int	必选	**参数解释：** 删除对象名个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
quiet	int	可选	**参数解释：** 指定是否使用quiet模式。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-223 obs_delete_object_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
delete_object_data_callback	**obs_delete_object_data_callback** *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-224 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 11-225 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-226 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 11-227 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-228 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-229 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-230 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 11-231 obs_object_info

参数名称	参数类型	是否必选	描述
key	char *	必选	**参数解释：** 对象名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
version_id	char *	可选	**参数解释：** 对象版本号，如果非多版本对象，请把version_id设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-232 obs_put_properties

参数名称	参数类型	是否必选	描述
content_type	char *	必选	**参数解释：** 指定对象被下载时的文件类型。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Type的取值。 **默认取值：** 无
md5	char *	必选	**参数解释：** 对象数据的MD5值（经过Base64编码），提供给OBS服务端，校验数据完整性。OBS服务端会将该MD5值与对象数据计算出的MD5值进行对比，如果不匹配则上传失败，返回HTTP 400错误。 **约束限制：** 对象数据的MD5值必须经过Base64编码。如果不设置对象的MD5值，OBS服务端会忽略对对象数据的MD5值校验。 **取值范围：** 按照RFC 1864标准计算出消息体的MD5摘要字符串，即消息体128-bit MD5值经过Base64编码后得到的字符串。 示例：n58IG6hfM7vqI4K0vnWpog== **默认取值：** 无
cache_control	char *	必选	**参数解释：** 指定对象被下载时的网页的缓存行为。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Cache-Control的取值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_disposition_filename	char *	必选	**参数解释：** 指定对象被下载时的名称。假设设置为test.txt则相当于添加了Content-Disposition: attachment; filename=test.txt头域 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Disposition的取值。 **默认取值：** 无
content_encoding	char *	必选	**参数解释：** 指定对象被下载时的内容编码格式。 **约束限制：** 无 **取值范围：** 参见HTTP标准头域Content-Encoding的取值。 **默认取值：** 无
website_redirect_location	char *	必选	**参数解释：** 当桶设置了Website配置，可以将获取这个对象的请求重定向到桶内另一个对象或一个外部的URL。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ **约束限制：** 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
get_conditions	**obs_get_conditions** *	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无
start_byte	uint64_t	可选	**参数解释：** 指定复制对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始复制）
byte_count	uint64_t	可选	**参数解释：** 指定复制的长度。 **约束限制：** • 取值必须大于0。 • 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
upload_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
expires	int64_t	可选	**参数解释：** OBS请求中Expires头的值。指定Object被下载时的网页的缓存过期时间。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_expires	int64_t	可选	**参数解释：** 指定对象过期时间，单位是天。过期之后对象会被自动删除。 **约束限制：** 设置的天数计算出的过期时间不能早于当前时间，如10天前上传的对象，不能设置小于10的值。 **取值范围：** 大于0的整数值。 **默认取值：** 无
canned_acl	**obs_canned_acl**	可选	**参数解释：** 权限控制策略。 **约束限制：** 无 **取值范围：** 请详见**obs_canned_acl**。
az_redundancy	**obs_az_redundancy**	可选	**参数解释：** 指定复制对象时的一系列参数。 **约束限制：** 无 **取值范围：** 请详见**obs_az_redundancy**。

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中元素的个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	**obs_name_value***	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制**： • 如果请求携带了此头域，那么响应的消息中应该包含此消息头。 • 所有自定义元数据大小的总和不超过8K。单个自定义元数据大小的计算方式为：每个键和值的UTF-8 编码中的字节总数。 • 自定义元数据的key值不区分大小写，OBS统一转为小写进行存储。value值区分大小写。 • 自定义元数据key-value对都必须符合US-ASCII。如果一定要使用非ASCII码或不可识别字符，需要客户端自行做编解码处理，可以采用URL编码或者Base64编码，服务端不会做解码处理。例如x-obs-meta-中文：中文经URL编码后发送，“中文”的URL编码为：%E4%B8%AD%E6%96%87，则响应为x-obs-meta-%E4%B8%AD%E6%96%87: %E4%B8%AD%E6%96%87
metadata_action	**metadata_action_indicator**	可选	**参数解释：** 元数据操作指示符。 **约束限制：** 无 **取值范围：** 请详见**metadata_action_indicator**。

参数名称	参数类型	是否必选	描述
server_callback	obs_upload_file_server_callback	可选	**参数解释：** 服务端回调相关参数。 **约束限制：** 无

表 11-233 obs_canned_acl

枚举值	说明
OBS_CANNED_ACL_PRIVATE	私有读写，桶或对象的所有者拥有完全控制的权限，其他任何人都没有访问权限。
OBS_CANNED_ACL_PUBLIC_READ	公共读私有写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、桶的多版本。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_WRITE	公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务。 设在对象上，所有人可以获取该对象内容和元数据。
OBS_CANNED_ACL_PUBLIC_READ_DELIVERED	桶公共读，桶内对象公共读，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据，可以获取该桶内对象的内容和元数据。 不能应用于对象。
OBS_CANNED_ACL_PUBLIC_READ_WRITE_DELIVERED	桶公共读写，桶内对象公共读写，设在桶上，所有人可以获取该桶内对象列表、桶内多段任务、桶的元数据、上传对象、删除对象、初始化段任务、上传段、合并段、复制段、取消多段上传任务，可以获取该桶内对象的内容和元数据。 不能应用于对象。

表 11-234 obs_get_conditions

参数名称	参数类型	是否必选	描述
start_byte	uint64_t	可选	**参数解释：** 指定下载对象的起始位置。 **约束限制：** 无 **取值范围：** [0~对象长度-1)，单位：字节。 **默认取值：** 0（即从对象的第一个字节开始下载）
byte_count	uint64_t	可选	**参数解释：** 指定下载的长度。 **约束限制：** ● 取值必须大于0。 ● 如果“start_byte+byte_count”大于对象长度-1，实际仍取对象长度-1，单位为字节。 **取值范围：** 无 **默认取值：** 无
download_limit	uint64_t	可选	**参数解释：** 对单链接请求的带宽限制。 **约束限制：** 无 **取值范围：** [819200，838860800]，单位 bit/s。 **默认取值：** 无
if_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
if_not_modified_since	int64_t	可选	**参数解释：** 如果对象在指定的时间后没有修改，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
if_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
if_not_match_etag	char *	可选	**参数解释：** 指定一个预设的Etag值，如果下载或复制对象的ETag值与该参数值不相同，则请求成功，否则返回错误。 **约束限制：** 无 **取值范围：** 长度为32的字符串。 **默认取值：** 无
image_process_config	**image_process_configure** *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无

表 11-235 metadata_action_indicator

枚举值	说明
OBS_NO_METADATA_ACTION	默认的无效值。
OBS_REPLACE	表示使用当前请求中携带的头域完整替换，未指定的元数据会被删除。
OBS_REPLACE_NEW	表示对于已经存在值的元数据进行替换，不存在值的元数据进行赋值，未指定的元数据保持不变（自定义元数据作替换处理）。

表 11-236 obs_upload_file_server_callback

参数名称	参数类型	是否必选	描述
callback_url	char *	必选	**参数解释：** 对象上传成功之后，OBS向此url发送回调请求，请求方法为POST。 支持设置多个url，以英文分号（;）分隔，最多支持10个。 callbackUrl需要做url编码。例如：“http://www.example.com/中文?key=中文名”需要编码成“http://www.example.com/%E4%B8%AD%E6%96%87?key=%E4%B8%AD%E6%96%87%E5%90%8D”。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_host	char *	可选	**参数解释：** 发起回调请求的Host头域的值，如果不设置，会使用callbackUrl解析出来的Host。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_body_type	char *	可选	**参数解释：** 发起回调请求的Content-Type头域的值。支持application/x-www-form-urlencoded、application/json。如果不设置，默认为application/json。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-237 image_process_configure

参数名称	参数类型	是否必选	描述
image_process_mode	image_process_mode_type	可选	**参数解释：** 图片处理参数头。 **约束限制：** 无 **取值范围：** 请详见image_process_mode_type。
cmds_stylename	char *	可选	**参数解释：** 图片处理相关参数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-238 image_process_mode_type

枚举值	说明
obs_image_process_invalid_mode	默认的无效值。
obs_image_process_cmd	图片处理参数以image开头。

枚举值	说明
obs_image_process_style	图片处理参数以style开头。

表 11-239 obs_delete_object_data_callback

参数名称	参数类型	是否必选	描述
contents_count	int	必选	**参数解释：** buffer的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
contents	obs_delete_objects *	必选	**参数解释：** 待上传的数据缓冲区，把要上传的数据复制到这个缓冲区。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 11-240 obs_delete_objects

参数名称	参数类型	是否必选	描述
key	constchar*	必选	**参数解释：** 每个删除结果的对象名。对象名是对象在桶中的完整路径，路径中不包含桶名。 例如，您对象的访问地址为examplebucket.obs.cn-north-4.myhuaweicloud.com/folder/test.txt 中，对象名为folder/test.txt。 **约束限制：** 同一个桶中存储的对象名必须是唯一的。 **取值范围：** 长度大于0且不超过1024的字符串。 **默认取值：** 无
code	constchar*	可选	**参数解释：** 删除失败结果的错误码。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
message	constchar*	可选	**参数解释：** 删除失败结果的错误消息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
delete_marker	constchar*	可选	**参数解释：** 当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，返回消息中该元素的值为true。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
delete_marker_version_id	constchar*	可选	**参数解释：** 请求创建或删除的删除标记版本号。 当批量删除请求访问的桶是多版本桶时，如果创建或删除一个删除标记，响应消息会返回该元素。该元素在以下两种情况中会出现： • 用户发送不带版本删除请求，即请求只有对象名，无版本号。这种情况下，系统会创建一个删除标记，并在响应中返回该删除标记的版本号。 • 用户发送带版本删除请求，即请求同时包含对象名以及版本号，但是该版本号标识一个删除标记。这种情况下，系统会删除此删除标记，并在响应中返回该删除标记的版本号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例

以下示例展示如何通过batch_delete_objects删除多个多版本对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
obs_status delete_objects_data_callback(int contentsCount,
```



```
    obs_delete_objects *delobjs,
    void *callbackData);
int main()
{
    // 以下示例展示如何通过batch_delete_objects删除多个多版本对象：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 要批量删除的对象信息
    obs_object_info objectinfo[100];
    objectinfo[0].key = "example_get_file_test_delete1";
    objectinfo[0].version_id = "G00111934EAA2CE900004016129AC991";
    objectinfo[1].key = "example_get_file_test_delete2";
    objectinfo[1].version_id = "G00111934EAA2CE900004016129AC992";
    obs_delete_object_info delobj;
    memset(&delobj, 0, sizeof(obs_delete_object_info));
    delobj.keys_number = 2;
    // 设置响应回调函数
    obs_delete_object_handler handler =
    {
        {&response_properties_callback, &response_complete_callback},
        &delete_objects_data_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 批量删除对象
    batch_delete_objects(&options, objectinfo, &delobj, 0, &handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("test batch_delete_objects successfully. \n");
    }
    else
    {
        printf("test batch_delete_objects faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
obs_status delete_objects_data_callback(int contentsCount,
    obs_delete_objects *delobjs,
    void *callbackData)
{
    int i;
    for (i = 0; i < contentsCount; i++) {
        const obs_delete_objects*content = &(delobjs[i]);
        printf("delete object result:\nobject key:%s\nerror code:%s\nerror message:%s\ndelete marker:%s
\ndelete marker version_id:%s\n",
                content->key, content->code, content->message, content->delete_marker, content-
>delete_marker_version_id);
    }
    return OBS_STATUS_OK;
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
```



```
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
void print_error_details(const obs_error_details *error) {
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
```



```
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void response_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    if (callback_data)
    {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    print_error_details(error);
}
```

12 生命周期管理(C SDK)

12.1 生命周期管理简介(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

OBS允许您对桶设置生命周期规则，实现自动转换对象的存储类别、自动淘汰过期的对象，以有效利用存储特性，优化存储空间。针对不同前缀的对象，您可以同时设置多条规则。一条规则包含

- 规则ID，用于标识一条规则，不能重复。
- 受影响的对象前缀，此规则只作用于符合前缀的对象。
- 最新版本对象的转换策略，指定方式为：
  a. 指定满足前缀的对象创建后第几天时转换为指定的存储类别。
  b. 直接指定满足前缀的对象转换为指定的存储类别的日期。
- 最新版本对象过期时间，指定方式为：
  a. 指定满足前缀的对象创建后第几天时过期。
  b. 直接指定满足前缀的对象过期日期。
- 历史版本对象转换策略，指定方式为：
  - 指定满足前缀的对象成为历史版本后第几天时转换为指定的存储类别。
- 历史版本对象过期时间，指定方式为：
  - 指定满足前缀的对象成为历史版本后第几天时过期。
- 是否生效标识。

更多关于生命周期的内容请参考**生命周期管理**。

说明

- 对象过期后会被OBS服务端自动删除。
- 对象转换策略中的时间必须早于对象过期时间；历史版本对象转换策略中的时间也必须早于历史版本对象的过期时间。
- 桶必须开启多版本状态，历史版本对象转换策略和历史版本对象过期时间配置才能生效。

12.2 设置桶的生命周期配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持用户配置指定的规则，实现定时删除桶中的对象或者定时转换对象的存储类别，从而节省存储费用，更多生命周期相关信息请参见**生命周期管理**。

调用设置桶的生命周期配置接口，您可以为指定桶设置生命周期策略。

接口约束

- 单个桶的生命周期规则条数没有限制，但一个桶中所有生命周期规则的XML描述总大小不能超过20KB。
- 您最多可以在一个桶下配置20条生命周期管理规则，超过20条将提示不支持。
- 您必须是桶拥有者或拥有设置桶的生命周期配置的权限，才能设置桶的生命周配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutLifecycleConfiguration权限，如果使用桶策略则需授予PutLifecycleConfiguration权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 对象生命周期到期以后，对象将会永久删除，无法恢复。
- 归档存储和深度归档存储不支持多AZ，因此不支持使用生命周期的存储类别转换功能，将多AZ的桶或对象转化为归档或深度归档存储。
- 低频访问存储的最低存储时间为30天，归档存储的最低存储时间为90天，深度归档存储的最低存储时间为180天。如果对象经过转换，归档存储时间少于最低存储时间，需要补足剩余天数的存储费用。

方法定义

```
void set_bucket_lifecycle_configuration(const obs_options *options,
        obs_lifecycle_conf* bucket_lifecycle_conf, unsigned int blcc_number,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 12-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
bucket_lifecycle_conf	**obs_lifecycle_conf** *	必选	**参数解释：** 桶生命周期配置说明，具体说明请参看下表。 **约束限制：** 无
blcc_number	unsigned int	必选	**参数解释：** 数组bucket_lifecycle_conf的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 12-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** 标准存储类别
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 12-4 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 12-5 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-6 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 12-7 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 12-8 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 12-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-11 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 12-12 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-13 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-14 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000

参数名称	参数类型	是否必选	描述
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-15 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	**obs_name_value***	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-16 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-17 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。

枚举值	说明
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 12-18 obs_lifecycle_conf

参数名称	参数类型	是否必选	描述
date	const char *	如果没有days元素，且没有transition, noncurrent_version_days, noncurrent_version_transition，则必选	**参数解释：** 表示针对最新版本的对象过期规则生效的时间。 **约束限制：** 该值必须兼容ISO8601格式，而且必须是UTC午夜0点。 **取值范围：** 无 **默认取值：** 无
days	const char *	如果没有date元素，且没有transition, noncurrent_version_days, noncurrent_version_transition，则必选	**参数解释：** 表示在对象创建时间后第几天时过期规则生效。 **约束限制：** 仅针对对象的最新版本。 **取值范围：** 无 **默认取值：** 无
id	const char *	可选	**参数解释：** 一条规则的标识。 **约束限制：** 无 **取值范围：** 由不超过255个字符的字符串组成。 **默认取值：** 无
prefix	const char *	必选	**参数解释：** 对象名前缀，用以标识哪些对象可以匹配到当前这条规则。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
status	const char *	必选	**参数解释：** 标识当前这条规则是否启用。 **约束限制：** 无 **取值范围：** • Enabled（代表启用） • Disabled（代表不启用） **默认取值：** 无
noncurrent_version_days	const char *	可选	**参数解释：** 生命周期配置中表示历史版本过期时间。您可以将该动作设置在已启用多版本（或暂停）的桶，来让系统删除对象的满足特定生命周期的历史版本。 **约束限制：** 仅针对历史版本。 **取值范围：** 无 **默认取值：** 无
transition	**obs_lifecycle_transtion** *	如果没有date，days，noncurrent_version_transition或者noncurrent_version_days，则必选	**参数解释：** 生命周期配置中表示迁移时间和迁移后对象存储级别的元素。 **约束限制：** 仅针对对象的最新版本。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
transition_num	unsigned int	如果transition非空，则必选	**参数解释：** 数组transition的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
noncurrent_version_transition	**obs_lifecycle_noncurrent_transtion***	如果没有date，days，transition或者noncurrent_version_days，则必选	**参数解释：** 生命周期配置中表示对象的历史版本迁移时间和迁移后对象存储级别的元素。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
noncurrent_version_transition_num	unsigned int	如果noncurrent_version_transition非空，则必选	**参数解释：** 数组noncurrent_version_transition的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-19 obs_lifecycle_transition

参数名称	参数类型	是否必选	描述
date	const char *	如果没有obs_lifecycle_transition.days元素，则必选	**参数解释：** 表示针对最新版本的对象过期规则生效的时间。 **约束限制：** 该值必须兼容ISO8601格式，而且必须是UTC午夜0点。 **取值范围：** 无 **默认取值：** 无
days	const char *	如果没有obs_lifecycle_transition.date元素，则必选	**参数解释：** 表示在对象创建时间后第几天时过期规则生效。 **约束限制：** 仅针对对象的最新版本。 **取值范围：** 无 **默认取值：** 无
storage_class	**obs_storage_class**	必选	**参数解释：** 表示最新版本对象将被修改成的存储级别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。

表 12-20 obs_lifecycle_noncurrent_transition

参数名称	参数类型	是否必选	描述
noncurrent_version_days	const char *	必选	**参数解释：** 表示对象在成为历史版本之后第几天时转换规则生效。 **约束限制：** 仅针对历史版本。 **取值范围：** 无 **默认取值：** 无
storage_class	**obs_storage_class**	必选	**参数解释：** 表示历史版本对象将被修改成的存储级别。 **约束限制：** 无 **取值范围：** 可详见**obs_storage_class**。

表 12-21 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 12-22 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-23 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-24 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-25 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释:** 错误响应消息体XML中的其他元素的值。 **约束限制:** 无
error_headers_count	int	**参数解释:** error_headers的头域数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
error_headers	char**	**参数解释:** 响应头域中包含error的所有头域。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

表 12-26 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释:** 属性的键。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-27 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例一：设置对象过期时间

以下示例展示如何设置最新版本对象和历史版本对象的过期时间：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置最新版本对象和历史版本对象的过期时间：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_lifecycle_conf bucket_lifecycle_conf;
    memset(&bucket_lifecycle_conf, 0, sizeof(obs_lifecycle_conf));
    //生命周期规则的id
    bucket_lifecycle_conf.id = "test1";
    // 指定前缀"test"
```



```
    bucket_lifecycle_conf.prefix = "test";
    // 指定满足前缀的对象创建10天后过期
    bucket_lifecycle_conf.days = "10";
    // 指定满足前缀的对象的历史版本20天后过期
    bucket_lifecycle_conf.noncurrent_version_days = "20";
    // 该生命周期规则生效
    bucket_lifecycle_conf.status = "Enabled";
    obs_status  ret_status = OBS_STATUS_BUTT;
    set_bucket_lifecycle_configuration(&options, &bucket_lifecycle_conf, 1,
        &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket lifecycle configuration success.\n");
    }
    else
    {
        printf("set bucket lifecycle configuration failed(%s).\n",
            obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
```



```
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例二：设置对象转换策略

以下示例展示如何设置最新版本对象和历史版本对象的转换策略：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置最新版本对象和历史版本对象的转换策略：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    obs_lifecycle_conf bucket_lifecycle_conf;
    memset(&bucket_lifecycle_conf, 0, sizeof(obs_lifecycle_conf));
    //生命周期规则的id
    bucket_lifecycle_conf.id = "test3";
    // 指定前缀"test"
    bucket_lifecycle_conf.prefix = "bcd";
    // 该生命周期规则生效
    bucket_lifecycle_conf.status = "Enabled";
    // 指定满足前缀的对象创建30天后过期
    bucket_lifecycle_conf.days = "30";
    obs_lifecycle_transtion transition;
    memset(&transition, 0, sizeof(obs_lifecycle_transtion));
    // 指定满足前缀的对象创建10天后转换
    transition.days = "10";
    // 指定对象转换后的存储类别
    transition.storage_class = OBS_STORAGE_CLASS_STANDARD_IA;
    bucket_lifecycle_conf.transition = &transition;
    bucket_lifecycle_conf.transition_num = 1;
    obs_lifecycle_noncurrent_transtion noncurrent_transition;
    memset(&noncurrent_transition, 0, sizeof(obs_lifecycle_noncurrent_transtion));
    // 指定满足前缀的对象的历史版本30天后转换
    noncurrent_transition.noncurrent_version_days = "30";
    // 指定满足前缀的对象的历史版本转换后的存储类别
    noncurrent_transition.storage_class = OBS_STORAGE_CLASS_STANDARD_IA;
    bucket_lifecycle_conf.noncurrent_version_transition = &noncurrent_transition;
    bucket_lifecycle_conf.noncurrent_version_transition_num = 1;
    obs_status  ret_status = OBS_STATUS_BUTT;
    set_bucket_lifecycle_configuration(&options, &bucket_lifecycle_conf, 1,
        &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket lifecycle configuration success.\n");
    }
    else
    {
        printf("set bucket lifecycle configuration failed(%s).\n",
            obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
```



```
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
```



```
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶的生命周期配置的API说明，请参见**设置桶的生命周期配置**。
- 更多关于设置桶的生命周期配置的代码示例，请参见**Github示例**。
- 设置桶的生命周期配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

12.3 获取桶的生命周期配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持用户配置指定的规则，实现定时删除桶中的对象或者定时转换对象的存储类别，从而节省存储费用，更多生命周期相关信息请参见**生命周期管理**。

调用获取桶的生命周期配置接口，您可以获取指定桶的生命周期策略。

接口约束

- 您必须是桶拥有者或拥有获取桶的生命周期配置的权限，才能获取桶的生命周配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetLifecycleConfiguration权限，如果使用桶策略则需授予GetLifecycleConfiguration权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_lifecycle_configuration(const obs_options *options,
                                        obs_lifecycle_handler *handler, void *callback_data);
```

请求参数说明

表 12-28 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_lifecycle_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-29 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 12-30 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 12-31 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 12-32 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-33 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 12-34 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 12-35 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 12-36 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-37 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-38 obs_lifecycle_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
get_lifecycle_callback	**get_lifecycle_configuration_callback** *	必选	**参数解释：** 回调函数指针，可以在这个回调中把桶生命周期规则的具体内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 12-39 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 12-40 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-41 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-42 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-43 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-44 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-45 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 12-46 get_lifecycle_configuration_callback

参数名称	参数类型	是否必选	描述
bucket_lifecycle_conf	**obs_lifecycle_conf** *	必选	**参数解释：** 桶生命周期规则具体内容。 **约束限制：** 无
blcc_number	unsigned int	必选	**参数解释：** bucket_lifecycle_conf的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-47 obs_lifecycle_conf

参数名称	参数类型	是否必选	描述
date	const char *	如果没有days元素，且没有transition, noncurrent_version_days, noncurrent_version_transition，则必选	**参数解释：** 表示针对最新版本的对象过期规则生效的时间。 **约束限制：** 该值必须兼容ISO8601格式，而且必须是UTC午夜0点。 **取值范围：** 无 **默认取值：** 无
days	const char *	如果没有date元素，且没有transition, noncurrent_version_days, noncurrent_version_transition，则必选	**参数解释：** 表示在对象创建时间后第几天时过期规则生效。 **约束限制：** 仅针对对象的最新版本。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
id	const char *	可选	**参数解释：** 一条Rule的标识。 **约束限制：** 无 **取值范围：** 由不超过255个字符的字符串组成。 **默认取值：** 无
prefix	const char *	必选	**参数解释：** 对象名前缀，用以标识哪些对象可以匹配到当前这条规则。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
status	const char *	必选	**参数解释：** 标识当前这条规则是否启用。 **约束限制：** 无 **取值范围：** ● Enabled（代表启用） ● Disabled（代表不启用） **默认取值：** 无

参数名称	参数类型	是否必选	描述
noncurrent_version_days	const char *	可选	**参数解释：** 生命周期配置中表示历史版本过期时间的Container。您可以将该动作设置在已启用多版本（或暂停）的桶，来让系统删除对象的满足特定生命周期的历史版本。 **约束限制：** 仅针对历史版本。 **取值范围：** 无 **默认取值：** 无
transition	**obs_lifecycle_transtion** *	如果没有date，days，noncurrent_version_transition或者noncurrent_version_days，则必选	**参数解释：** 生命周期配置中表示迁移时间和迁移后对象存储级别的元素。 **约束限制：** 仅针对对象的最新版本。
transition_num	unsigned int	如果transition非空，则必选	**参数解释：** 数组transition的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
noncurrent_version_transition	**obs_lifecycle_noncurrent_transtion***	如果没有date，days，transition或者noncurrent_version_days，则必选	**参数解释：** 生命周期配置中表示对象的历史版本迁移时间和迁移后对象存储级别的元素。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
noncurrent_version_transition_num	unsigned int	如果noncurrent_version_transition非空，则必选	**参数解释：** 数组noncurrent_version_transition的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-48 obs_lifecycle_transition

参数名称	参数类型	是否必选	描述
date	const char *	如果没有obs_lifecycle_transition.days元素，则必选	**参数解释：** 表示针对最新版本的对象过期规则生效的时间。 **约束限制：** 该值必须兼容ISO8601格式，而且必须是UTC午夜0点。 **取值范围：** 无 **默认取值：** 无
days	const char *	如果没有obs_lifecycle_transition.date元素，则必选	**参数解释：** 表示在对象创建时间后第几天时过期规则生效。 **约束限制：** 仅针对对象的最新版本。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	必选	**参数解释：** 表示最新版本对象将被修改成的存储级别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。

表 12-49 obs_lifecycle_noncurrent_transition

参数名称	参数类型	是否必选	描述
noncurrent_version_days	const char *	必选	**参数解释：** 表示对象在成为历史版本之后第几天时转换规则生效。 **约束限制：** 仅针对历史版本。 **取值范围：** 无 **默认取值：** 无
storage_class	obs_storage_class	必选	**参数解释：** 表示历史版本对象将被修改成的存储级别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。

代码示例：获取桶的生命周期配置

以下示例展示如何获取桶的生命周期规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status getBucketLifecycleConfigurationCallbackEx(obs_lifecycle_conf* bucketLifeCycleConf,
    unsigned int blccNumber,
    void *callback_data);
int main()
{
    // 以下示例展示如何查看桶的生命周期规则：
```

```
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    // 设置回调函数
    obs_lifecycle_handler lifeCycleHandlerEx =
    {
        response_handler,
        &getBucketLifecycleConfigurationCallbackEx
    };
    obs_status  ret_status = OBS_STATUS_BUTT;
    get_bucket_lifecycle_configuration(&options, &lifeCycleHandlerEx, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("get_lifecycle_config success.\n");
    }
    else
    {
        printf("get_lifecycle_config faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
```



```
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
obs_status getBucketLifecycleConfigurationCallbackEx(obs_lifecycle_conf* bucketLifeCycleConf,
    unsigned int blccNumber,
    void *callback_data)
{
    unsigned int i = 0;
#define print_nonull(name, field)                               \
    do {                                                \
```

```
        if (field && field[0]) {                                \
            printf("%s: %s\n", name, field);          \
        }                                                       \
    } while (0)
    for (i = 0; i < blccNumber; i++)
    {
        printf("----------------------------------------------\n");
        print_nonull("id", bucketLifeCycleConf[i].id);
        print_nonull("prefix", bucketLifeCycleConf[i].prefix);
        print_nonull("status", bucketLifeCycleConf[i].status);
        print_nonull("days", bucketLifeCycleConf[i].days);
        print_nonull("date", bucketLifeCycleConf[i].date);
    }
    printf("----------------------------------------------\n");
    return OBS_STATUS_OK;
}
```

12.4 删除桶的生命周期配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS支持用户配置指定的规则，实现定时删除桶中的对象或者定时转换对象的存储类别，从而节省存储费用，更多生命周期相关信息请参见**生命周期管理**。

调用删除桶的生命周期配置接口，您可以删除指定桶的生命周期策略。

接口约束

- 您必须是桶拥有者或拥有删除桶的生命周期配置的权限，才能删除桶的生命周配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutLifecycleConfiguration权限，如果使用桶策略则需授予PutLifecycleConfiguration权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void delete_bucket_lifecycle_configuration(const obs_options *options,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 12-50 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-51 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 12-52 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_list_type**。

表 12-53 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 12-54 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-55 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 12-56 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 12-57 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 12-58 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 12-59 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-60 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 12-61 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-62 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 12-63 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-64 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-65 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 12-66 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：删除桶的生命周期配置

以下示例展示如何删除桶的生命周期规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status getBucketLifecycleConfigurationCallbackEx(obs_lifecycle_conf* bucketLifeCycleConf,
    unsigned int blccNumber,
    void *callback_data);
int main()
{
    // 以下示例展示如何删除桶的生命周期规则：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status  ret_status = OBS_STATUS_BUTT;
    delete_bucket_lifecycle_configuration(&options, &response_handler, &ret_status);
```

```
    if (OBS_STATUS_OK == ret_status) {
        printf("test_delete_lifecycle_config success.\n");
    }
    else
    {
        printf("test_delete_lifecycle_config faied(%s).\n",
            obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
```



```
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于删除桶的生命周期配置的API说明，请参见**删除桶的生命周期配置**。
- 更多关于删除桶的生命周期配置的代码示例，请参见**Github示例**。
- 删除桶的生命周期配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

13 跨域资源共享(C SDK)

13.1 设置桶的 CORS 配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

跨域资源共享（Cross Origin Resource Sharing，CORS）是由W3C标准化组织提出的一种网络浏览器的规范机制，定义了一个域中加载的客户端Web应用程序与另一个域中的资源交互的方式。而在通常的网页请求中，由于同源安全策略（Same Origin Policy，SOP）的存在，不同域之间的网站脚本和内容是无法进行交互的。OBS支持CORS规范，允许跨域请求访问OBS中的资源。

调用设置桶的CORS配置接口，您可设置指定桶的跨域资源共享规则，以允许客户端浏览器进行跨域请求。

接口约束

- 您必须是桶拥有者或拥有设置桶的CORS配置的权限，才能设置桶的CORS配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketCORS权限，如果使用桶策略则需授予PutBucketCORS权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_cors_configuration(const obs_options *options, obs_bucket_cors_conf *obs_cors_conf_info,
        unsigned int conf_num, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 13-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
obs_cors_conf_info	**obs_bucket_cors_conf** *	必选	**参数解释：** CORS规则具体内容。 **约束限制：** 无
conf_num	unsigned int	必选	**参数解释：** 数组obs_cors_conf_info的数组成员个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 13-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 13-4 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 13-5 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 13-6 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 13-7 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 13-8 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-11 obs_bucket_cors_conf

参数名称	参数类型	是否必选	描述
id	const char *	可选	**参数解释：** 一条CORS规则的标识。 **约束限制：** 无 **取值范围：** 1~255个字符的字符串组成。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allowed_method	const char **	必选	**参数解释** 指定允许的跨域请求HTTP方法，即桶和对象的几种操作类型。 **约束限制：** 无 **取值范围：** 支持以下HTTP方法： ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无
allowed_method_number	unsigned int	可选	**参数解释** allowed_method的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allowed_origin	const char **	可选	**参数解释：** 指定允许的跨域请求的来源，即允许来自该域名下的请求访问该桶。 **约束限制：** 表示域名的字符串，仅支持英文域名。通过正则表达式进行匹配，每个匹配规则允许使用最多一个“*”通配符。例如：https://*.vbs.example.com。 **取值范围：** 符合CORS协议的取值范围，长度为[0-20480]个字符。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allowed_origin_number	unsigned int	可选	**参数解释** allowed_origin的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allowed_header	const char **	可选	**参数解释：** 指定允许的跨域请求的头域。配置CORS请求中允许携带的“Access-Control-Request-Headers”头域。如果一个请求带了“Access-Control-Request-Headers”头域，则只有匹配上AllowedHeader中的配置才认为是一个合法的CORS请求（通过正则表达式进行匹配）。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围，长度为[0-20480]个字符。 **默认取值：** 无
allowed_header_number	unsigned int	可选	**参数解释** allowed_header的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
max_age_seconds	const char *	可选	**参数解释：** 请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个max_age_seconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
expose_header	const char **	可选	**参数解释：** CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
expose_header_number	unsigned int	可选	**参数解释** expose_header的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-12 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 13-13 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-14 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 13-15 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-16 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-17 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-18 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置桶的 CORS 配置

以下示例展示如何设置桶的跨域规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置桶的跨域规则：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    char *id_1 = "1";
    // 指定浏览器对特定资源的预取(OPTIONS)请求返回结果的缓存时间,单位为秒
    char *max_age_seconds = "100";
    // 指定允许的跨域请求方法(GET/PUT/DELETE/POST/HEAD)
```



```
    const char* allowedMethod_1[5] = { "GET","PUT","HEAD","POST","DELETE" };
    // 指定允许跨域请求的来源
    const char* allowedOrigin_1[2] = { "obs.example.com", "www.example.com" };
    // 指定允许用户从应用程序中访问的header
    const char* allowedHeader_1[2] = { "header-1", "header-2" };
    // 响应中带的附加头域
    const char* exposeHeader_1[2] = { "hello", "world" };
    obs_bucket_cors_conf bucketCorsConf;
    memset(&bucketCorsConf, 0, sizeof(obs_bucket_cors_conf));
    bucketCorsConf.id = id_1;
    bucketCorsConf.max_age_seconds = max_age_seconds;
    bucketCorsConf.allowed_method = allowedMethod_1;
    bucketCorsConf.allowed_method_number = 5;
    bucketCorsConf.allowed_origin = allowedOrigin_1;
    bucketCorsConf.allowed_origin_number = 2;
    bucketCorsConf.allowed_header = allowedHeader_1;
    bucketCorsConf.allowed_header_number = 2;
    bucketCorsConf.expose_header = exposeHeader_1;
    bucketCorsConf.expose_header_number = 2;
    set_bucket_cors_configuration(&options, &bucketCorsConf, 1, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set_bucket_cors success.\n");
    }
    else {
        printf("set_bucket_cors faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
```



```
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

说明

allowed_origin、allowed_method、allowed_header都能够最多支持一个通配符“*”。“*”表示对于所有的域来源、操作或者头域都满足。

相关链接

- 关于设置桶的CORS配置的API说明，请参见**设置桶的CORS配置**。
- 更多关于设置桶的CORS配置的代码示例，请参见**Github示例**。
- 设置桶的CORS配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

13.2 获取桶的 CORS 配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

跨域资源共享（Cross Origin Resource Sharing，CORS）是由W3C标准化组织提出的一种网络浏览器的规范机制，定义了一个域中加载的客户端Web应用程序与另一个域中的资源交互的方式。而在通常的网页请求中，由于同源安全策略（Same Origin Policy，SOP）的存在，不同域之间的网站脚本和内容是无法进行交互的。OBS支持CORS规范，允许跨域请求访问OBS中的资源。

调用获取桶的CORS配置接口，您可获取指定桶的跨域资源共享规则。

接口约束

- 您必须是桶拥有者或拥有获取桶的CORS配置的权限，才能获取桶的CORS配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketCORS权限，如果使用桶策略则需授予GetBucketCORS权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_cors_configuration(const obs_options *options, obs_cors_handler *handler,
        void *callback_data);
```

请求参数说明

表 13-19 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	obs_cors_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-20 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 13-21 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见obs_bucket_list_type。

表 13-22 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 13-23 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000

参数名称	参数类型	是否必选	描述
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-24 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 13-25 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 13-26 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 13-27 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-28 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-29 obs_cors_handler

参数名称	参数类型	是否必选	描述
response_handler	obs_response_handler *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
get_cors_callback	get_cors_configuration_callback *	必选	**参数解释：** 标签回调函数指针，可以在这个回调中把CORS规则具体内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 13-30 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 13-31 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties *	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-32 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-33 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-34 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-35 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-36 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 13-37 get_cors_configuration_callback

参数名称	参数类型	是否必选	描述
bucket_cors_conf	**obs_bucket_cors_conf** *	必选	**参数解释：** CORS规则具体内容。 **约束限制：** 无
bcc_number	unsigned int	必选	**参数解释：** bucket_cors_conf的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-38 obs_bucket_cors_conf

参数名称	参数类型	是否必选	描述
id	const char *	可选	**参数解释：** 一条CORS规则的标识。 **约束限制：** 无 **取值范围：** 1~255个字符的字符串组成。 **默认取值：** 无
allowed_method	const char **	必选	**参数解释** 指定允许的跨域请求HTTP方法，即桶和对象的几种操作类型。 **约束限制：** 无 **取值范围：** 支持以下HTTP方法： ● GET ● PUT ● HEAD ● POST ● DELETE **默认取值：** 无

参数名称	参数类型	是否必选	描述
allowed_method_number	unsigned int	可选	**参数解释** allowed_method的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allowed_origin	const char **	可选	**参数解释：** 指定允许的跨域请求的来源，即允许来自该域名下的请求访问该桶。 **约束限制：** 表示域名的字符串，仅支持英文域名。通过正则表达式进行匹配，每个匹配规则允许使用最多一个“*”通配符。例如：https://*.vbs.example.com。 **取值范围：** 符合CORS协议的取值范围，长度为[0-20480]个字符。 **默认取值：** 无
allowed_origin_number	unsigned int	可选	**参数解释** allowed_origin的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allowed_header	const char **	可选	**参数解释：** 指定允许的跨域请求的头域。配置CORS请求中允许携带的“Access-Control-Request-Headers”头域。如果一个请求带了“Access-Control-Request-Headers”头域，则只有匹配上AllowedHeader中的配置才认为是一个合法的CORS请求（通过正则表达式进行匹配）。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围，长度为[0-20480]个字符。 **默认取值：** 无
allowed_header_number	unsigned int	可选	**参数解释** allowed_header的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
max_age_seconds	const char *	可选	**参数解释：** 请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个max_age_seconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000

参数名称	参数类型	是否必选	描述
expose_header	const char **	可选	**参数解释：** CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
expose_header_number	unsigned int	可选	**参数解释** expose_header的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：获取桶的 CORS 配置

以下示例展示如何获取桶的跨域规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status get_cors_info_callback(obs_bucket_cors_conf* bucket_cors_conf,
    unsigned int bcc_number,
    void *callback_data);
int main()
{
    // 以下示例展示如何查看桶的跨域规则：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
```



```
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 设置回调函数
    obs_cors_handler cors_handler_info =
    {
        response_handler,
        &get_cors_info_callback
    };
    get_bucket_cors_configuration(&options, &cors_handler_info, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("get_cors_config success.\n");
    }
    else {
        printf("get_cors_config faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
```



```
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
obs_status get_cors_info_callback(obs_bucket_cors_conf* bucket_cors_conf,
    unsigned int bcc_number,
    void *callback_data)
{
    unsigned int i = 0;
    for (i = 0; i < bcc_number; ++i)
    {
        printf("-------------------------------------------------------\n");
        printf("id = %s\nmaxAgeSeconds = %s\n", bucket_cors_conf[i].id,
bucket_cors_conf[i].max_age_seconds);
        unsigned int j;
        for (j = 0; j < bucket_cors_conf[i].allowed_method_number; j++)
        {
            printf("allowedMethodes = %s\n", bucket_cors_conf[i].allowed_method[j]);
        }
        for (j = 0; j < bucket_cors_conf[i].allowed_origin_number; j++)
        {
            printf("allowedOrigines = %s\n", bucket_cors_conf[i].allowed_origin[j]);
        }
```



```
        for (j = 0; j < bucket_cors_conf[i].allowed_header_number; j++)
        {
            printf("allowedHeaderes = %s\n", bucket_cors_conf[i].allowed_header[j]);
        }
        for (j = 0; j < bucket_cors_conf[i].expose_header_number; j++)
        {
            printf("exposeHeaderes = %s\n", bucket_cors_conf[i].expose_header[j]);
        }
    }
    printf("-------------------------------------------------------\n");
    return OBS_STATUS_OK;
}
```

相关链接

- 关于获取桶的CORS配置的API说明，请参见**获取桶的CORS配置**。
- 更多关于获取桶的CORS配置的代码示例，请参见**Github示例**。
- 获取桶的CORS配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

13.3 删除桶的 CORS 配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

跨域资源共享（Cross Origin Resource Sharing，CORS）是由W3C标准化组织提出的一种网络浏览器的规范机制，定义了一个域中加载的客户端Web应用程序与另一个域中的资源交互的方式。而在通常的网页请求中，由于同源安全策略（Same Origin Policy，SOP）的存在，不同域之间的网站脚本和内容是无法进行交互的。OBS支持CORS规范，允许跨域请求访问OBS中的资源。

调用删除桶的CORS配置接口，您可删除指定桶的跨域资源共享规则。

接口约束

- 您必须是桶拥有者或拥有删除桶的CORS配置的权限，才能删除桶的CORS配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketCORS权限，如果使用桶策略则需授予PutBucketCORS权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void delete_bucket_cors_configuration(const obs_options *options,
                                      obs_response_handler *handler, void *callback_data);
```


请求参数说明

表 13-39 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-40 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	temp_auth_configure*	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 13-41 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过obs_protocol控制是使用http协议还是https协议。 **取值范围：** 您可以从这里查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 可详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 可详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 可详见**obs_bucket_list_type**。

表 13-42 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 13-43 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-44 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 13-45 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 13-46 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 13-47 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 13-48 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-49 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 13-50 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-51 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 可详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 13-52 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-53 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-54 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 13-55 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：删除桶的 CORS 配置

以下示例展示如何删除桶的跨域规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status get_cors_info_callback(obs_bucket_cors_conf* bucket_cors_conf,
    unsigned int bcc_number,
    void *callback_data);
int main()
{
    // 以下示例展示如何删除桶的跨域规则：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    delete_bucket_cors_configuration(&options, &response_handler, &ret_status);
```



```
    if (OBS_STATUS_OK == ret_status) {
        printf("delete_cors_config success.\n");
    }
    else
    {
        printf("delete_cors_config faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                  \
        if (properties-> field) {                             \
            printf("%s: %s\n", name, properties->field);          \
        }                                                 \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
```



```
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于删除桶的CORS配置的API说明，请参见**删除桶的CORS配置**。
- 更多关于删除桶的CORS配置的代码示例，请参见**Github示例**。
- 删除桶的CORS配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

14 设置访问日志(C SDK)

14.1 设置桶日志管理配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

调用设置桶日志管理配置接口，您可以为指定桶打开桶日志功能，并配置日志存放的目标桶。创建桶时，默认是不生成桶的日志的，调用C SDK的设置桶日志管理配置接口时，接口会自动为您打开桶日志配置。桶日志功能开启后，桶的每次操作将会产生一条日志，并将多条日志打包成一个日志文件。日志文件存放位置需要在开启桶日志功能时指定，可以存放到开启日志功能的桶中，也可以存放到其他您有权限的桶中，但需要和开启日志功能的桶在同一个region中。您还可以根据需要配置日志文件的访问权限，以及日志文件的文件名前缀。

接口约束

- 由于日志文件是OBS产生，并且由OBS上传到存放日志的桶中，因此OBS需要获得委托授权，用于上传生成的日志文件，所以在配置桶日志管理前，需要先到统一身份认证服务生成一个对OBS服务的委托，委托配置权限只需设置目标桶的上传对象权限。如何创建委托请参考**创建云服务委托**。
- 您必须是桶拥有者或拥有设置桶日志管理配置的权限，才能设置桶日志管理配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketLogging权限，如果使用桶策略则需授予PutBucketLogging权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_logging_configuration_obs(const obs_options *options, char *target_bucket,
        char *target_prefix, char *agency, obs_acl_group *acl_group,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 14-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
target_bucket	char *	必选	**参数解释：** 在生成日志时，源桶的所有者可以指定一个目标桶，将生成的所有日志放到该桶中。 在OBS系统中，支持多个源桶生成的日志放在同一个目标桶中，如果这样做，就需要指定不同的target_prefix以达到为来自不同源桶的日志分类的目的。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
target_prefix	char *	必选	**参数解释：** 通过该元素可以指定一个前缀给一类日志生成的对象。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
agency	char *	设置logging时必选，关闭logging时勿选。	**参数解释：** 产生logging日志桶owner创建委托OBS上传logging日志的委托名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_group	**obs_acl_group** *	可选	**参数解释：** 权限信息组结构体。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 14-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 14-4 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。

枚举值	说明
OBS_PROTOCOL_HTTP	使用http协议访问。

表 14-5 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 14-6 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 14-7 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 14-8 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-11 obs_acl_group

参数名称	参数类型	是否必选	描述
acl_grant_count	int	可选	**参数解释** acl_grants数组的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grants	**obs_acl_grant** *	可选	**参数解释** 权限信息结构体。 **约束限制：** 无

表 14-12 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	**obs_grantee_type**	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见**obs_grantee_type**。
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 14-13 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 14-14 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 14-15 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

表 14-16 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback*	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 14-17 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-18 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-19 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000

参数名称	参数类型	是否必选	描述
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。

参数名称	参数类型	是否必选	描述
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-20 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无

参数名称	参数类型	描述
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details	**obs_name_value***	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无

参数名称	参数类型	描述
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-21 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-22 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。

枚举值	说明
OBS_STATUS_SlowDown	请求频率过快。

代码示例一：开启桶日志

以下示例展示如何开启桶日志：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何开启桶日志：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 目标桶Owner通过统一身份认证服务创建的对OBS服务的委托的名称。
    char *agency = "storageSDK";
    //设置日志存储的目标桶名
    char * bucket_name_target = "example-dest-bucket-name";
    // 设置桶日志配置
    set_bucket_logging_configuration_obs(&options, bucket_name_target, "prefix-log", agency,
        NULL, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set bucket (%s) logging successfully. \n", bucketName);
    }
    else
    {
        printf("set bucket logging faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
```



```
            data->buffer, data->buffer_len);
        return OBS_STATUS_OK;
    }
    else {
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
```



```
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例二：为日志对象设置权限

以下示例展示如何为日志对象设置权限：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何为日志对象设置权限：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    int aclGrantCount = 2;
    obs_acl_grant acl_grants[2] = { 0 };
    // 为OBS授权用户设置对日志文件的全部权限
    acl_grants[0].grantee_type = OBS_GRANTEE_TYPE_CANONICAL_USER;
    strcpy(acl_grants[0].grantee.canonical_user.id, "userid1");
    strcpy(acl_grants[0].grantee.canonical_user.display_name, "dis1");
    acl_grants[0].permission = OBS_PERMISSION_READ;
    // 为所有用户设置对日志对象的读权限
    acl_grants[1].grantee_type = OBS_GRANTEE_TYPE_ALL_OBS_USERS;
    acl_grants[1].permission = OBS_PERMISSION_READ;
    obs_acl_group acl_group;
    acl_group.acl_grants = acl_grants;
    acl_group.acl_grant_count = aclGrantCount;
    // 目标桶Owner通过统一身份认证服务创建的对OBS服务的委托的名称。
    char *agency = "storageSDK";
    //设置日志存储的目标桶名
    char * bucket_name_target = "example-dest-bucket-name";
    // 设置桶日志配置
    // 设置日志文件前缀，也就是存储的文件夹名称，如不设置默认存储到目标桶的根目录下
    set_bucket_logging_configuration_obs(&options, bucket_name_target, "prefix-log", agency,
```



```
        &acl_group, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("set bucket(%s) logging successfully. \n", bucketName);
    }
    else
    {
        printf("set bucket logging faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
```



```
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例三：关闭桶日志

以下示例展示如何关闭桶日志：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示关闭桶日志：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
```



```
&response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    char * agency = NULL;
    char * bucket_name_target = NULL;
    // 设置桶日志配置
    set_bucket_logging_configuration_obs(&options, bucket_name_target, NULL, agency,
        NULL, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("close bucket(%s) logging successfully. \n", bucketName);
    }
    else
    {
        printf("close bucket logging faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
```



```
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶日志管理配置的API说明，请参见**设置桶日志管理配置**。
- 更多关于设置桶日志管理配置的代码示例，请参见**Github示例**。
- 设置桶日志管理配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

14.2 获取桶日志管理配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

调用获取桶日志管理配置，可获取指定桶的日志配置。

接口约束

- 您必须是桶拥有者或拥有获取桶日志管理配置的权限，才能获取桶日志管理配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketLogging权限，如果使用桶策略则需授予GetBucketLogging权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_logging_configuration(const obs_options *options, obs_response_handler *handler,
        bucket_logging_message *logging_message_data, void *callback_data);
```

请求参数说明

表 14-23 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
logging_message_data	**bucket_logging_message** *	必选	**参数解释：** 当前桶日志管理配置情况。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-24 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 14-25 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** • 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 • 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 • 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 14-26 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 14-27 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 14-28 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 14-29 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 14-30 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-31 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-32 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-33 bucket_logging_message

参数名称	参数类型	是否必选	描述
target_bucket	char *	可选	**参数解释** 在生成日志时，源桶的所有者可以指定一个目标桶，将生成的所有日志放到该桶中。 在OBS系统中，支持多个源桶生成的日志放在同一个目标桶中，如果这样做，就需要指定不同的target_prefix以达到为来自不同源桶的日志分类的目的。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
target_bucket_size	int	可选	**参数解释** 表示target_bucket数组的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
target_prefix	char *	可选	**参数解释** 通过该元素可以指定一个前缀给一类日志生成的对象。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
target_prefix_size	int	可选	**参数解释** 表示target_prefix数组的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
acl_grants	**obs_acl_grant** *	可选	**参数解释** 权限信息结构体。 **约束限制：** 无
acl_grant_count	int	可选	**参数解释** 表示acl_grants数组的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
agency	char *	可选	**参数解释** 产生logging日志桶owner创建委托OBS上传logging日志的委托名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
agency_size	int	可选	**参数解释** 表示agency数组的元素数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-34 obs_acl_grant

参数名称	参数类型	是否必选	描述
grantee_type	**obs_grantee_type**	必选	**参数解释：** 授权者类型描述。 **约束限制：** 无 **取值范围：** 请详见**obs_grantee_type**。
grantee.canonical_user.id	char[]	可选	**参数解释：** 被授权用户的ID，即user_id。 **约束限制：** 无 **取值范围：** 如何获取被授权用户的ID请参见**如何获取账号ID和用户ID?** **默认取值：** 无
permission	**obs_permission**	必选	**参数解释：** 桶访问权限。 **约束限制：** 无 **取值范围：** 请详见**obs_permission**。
bucket_delivered	**obs_bucket_delivered**	必选	**参数解释：** 桶的ACL是否向桶内对象传递。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_delivered**。 **默认取值：** BUCKET_DELIVERED_FALSE（指对象ACL不继承桶的ACL）

表 14-35 obs_grantee_type

枚举值	说明
OBS_GRANTEE_TYPE_CANONICAL_USER	OBS用户，桶或对象的权限可以授予任何拥有OBS账户的用户，被授权后对应的 OBS 用户可以使用AK和SK访问OBS。
OBS_GRANTEE_TYPE_ALL_USERS	所有人，所有人都可以访问对应的桶或对象，包括匿名用户。

表 14-36 obs_permission

枚举值	说明
OBS_PERMISSION_READ	读权限，如果有桶的读权限，则可以获取该桶内对象列表和桶的元数据。 如果有对象的读权限，则可以获取该对象内容和元数据。
OBS_PERMISSION_WRITE	写权限，如果有桶的写权限，则可以上传、覆盖和删除该桶内任何对象。 此权限在对象上不适用。
OBS_PERMISSION_READ_ACP	读ACP权限，如果有读ACP的权限，则可以获取对应的桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有读对应桶或对象ACP的权限。
OBS_PERMISSION_WRITE_ACP	写ACP权限，如果有写ACP的权限，则可以更新对应桶或对象的权限控制列表（ACL）。 桶或对象的所有者永远拥有写对应桶或对象的ACP的权限。 拥有了写ACP的权限，由于可以更改权限控制策略，实际上意味着拥有了完全访问的权限。
OBS_PERMISSION_FULL_CONTROL	完全控制权限，如果有桶的完全控制权限意味着拥有READ、WRITE、READ_ACP和WRITE_ACP的权限。 如果有对象的完全控制权限意味着拥有READ、READ_ACP和WRITE_ACP的权限。

表 14-37 obs_bucket_delivered

枚举值	说明
BUCKET_DELIVERED_FALSE	桶的ACL不向桶内对象传递。
BUCKET_DELIVERED_TRUE	桶的ACL向桶内对象传递。

表 14-38 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 14-39 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 14-40 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。
error_details	const obs_error_details*	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

表 14-41 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-42 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-43 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 14-44 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：获取桶日志管理配置

以下示例展示如何获取桶日志管理配置：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
void init_bucket_get_logging_message(bucket_logging_message *logging_message);
void print_grant_info(int acl_grant_count, obs_acl_grant *acl_grants);
void destroy_logging_message(bucket_logging_message *logging_message);
int main()
{
    // 以下示例展示如何查看桶日志配置：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 初始化日志配置结构体
```

```
    bucket_logging_message logging_message;
    init_bucket_get_logging_message(&logging_message);
    // 获取桶日志配置
    get_bucket_logging_configuration(&options, &response_handler, &logging_message, &ret_status);
    if (OBS_STATUS_OK == ret_status)
    {
        if (logging_message.target_bucket)
        {
            printf("Target_Bucket: %s\n", logging_message.target_bucket);
            if (logging_message.target_prefix)
            {
                printf("Target_Prefix: %s\n", logging_message.target_prefix);
            }
            if (logging_message.agency && logging_message.agency[0] != '\0')
            {
                printf("Agency: %s\n", logging_message.agency);
            }
            // 打印ACL配置
            print_grant_info(*logging_message.acl_grant_count, logging_message.acl_grants);
        }
        else
        {
            printf("Service logging is not enabled for this bucket.\n");
        }
    }
    else
    {
        printf("Get bucket logging faied(%s).\n", obs_get_status_name(ret_status));
    }
    // 销毁日志配置结构体
    destroy_logging_message(&logging_message);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("Error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("Error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
```



```
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    } else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                   error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void init_bucket_get_logging_message(bucket_logging_message *logging_message)
{
    memset(logging_message, 0, sizeof(bucket_logging_message));
    logging_message->target_bucket = (char *)malloc(sizeof(char)*OBS_MAX_HOSTNAME_SIZE);
    memset(logging_message->target_bucket, 0, OBS_MAX_HOSTNAME_SIZE);
    logging_message->target_bucket_size = OBS_MAX_HOSTNAME_SIZE;
    logging_message->target_prefix = (char *)malloc(sizeof(char)*OBS_MAX_KEY_SIZE);
    memset(logging_message->target_prefix, 0, OBS_MAX_KEY_SIZE);
    logging_message->target_prefix_size = OBS_MAX_KEY_SIZE;
```



```
    logging_message->agency = (char *)malloc(sizeof(char)*OBS_MAX_KEY_SIZE);
    memset(logging_message->agency, 0, OBS_MAX_KEY_SIZE);
    logging_message->agency_size = OBS_MAX_KEY_SIZE;
    logging_message->acl_grants =
(obs_acl_grant*)malloc(sizeof(obs_acl_grant)*OBS_MAX_ACL_GRANT_COUNT);
    memset(logging_message->acl_grants, 0, sizeof(obs_acl_grant)*OBS_MAX_ACL_GRANT_COUNT);
    logging_message->acl_grant_count = (int *)malloc(sizeof(int));
    *(logging_message->acl_grant_count) = 0;
}
void destroy_logging_message(bucket_logging_message *logging_message)
{
    free(logging_message->target_bucket);
    free(logging_message->target_prefix);
    free(logging_message->acl_grants);
    free(logging_message->agency);
    free(logging_message->acl_grant_count);
}
void print_grant_info(int acl_grant_count, obs_acl_grant *acl_grants)
{
    int i;
    for (i = 0; i < acl_grant_count; i++)
    {
        obs_acl_grant *grant = acl_grants + i;
        const char *type;
        char composedId[OBS_MAX_GRANTEE_USER_ID_SIZE +
            OBS_MAX_GRANTEE_DISPLAY_NAME_SIZE + 16] = { 0 };
        const char *id;
        switch (grant->grantee_type) {
        case OBS_GRANTEE_TYPE_HUAWEI_CUSTOMER_BYEMAIL:
            type = "Email";
            id = grant->grantee.huawei_customer_by_email.email_address;
            break;
        case OBS_GRANTEE_TYPE_CANONICAL_USER:
            type = "UserID";
            snprintf(composedId, sizeof(composedId),
                "%s (%s)", grant->grantee.canonical_user.id,
                grant->grantee.canonical_user.display_name);
            id = composedId;
            break;
        case OBS_GRANTEE_TYPE_ALL_OBS_USERS:
            type = "Group";
            id = "Authenticated Users";
            break;
        default:
            type = "Group";
            id = "All Users";
            break;
        }
        const char *perm;
        switch (grant->permission) {
        case OBS_PERMISSION_READ:
            perm = "READ";
            break;
        case OBS_PERMISSION_WRITE:
            perm = "WRITE";
            break;
        case OBS_PERMISSION_READ_ACP:
            perm = "READ_ACP";
            break;
        case OBS_PERMISSION_WRITE_ACP:
            perm = "WRITE_ACP";
            break;
        default:
            perm = "FULL_CONTROL";
            break;
        }
        printf("%-6s  %-90s  %-12s\n", type, id, perm);
    }
}
```


相关链接

- 关于获取桶日志管理配置的API说明，请参见**获取桶日志管理配置**。
- 更多关于获取桶日志管理配置的代码示例，请参见**Github示例**。
- 获取桶日志管理配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

15 静态网站托管(C SDK)

15.1 网站文件托管(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

您可通过以下步骤实现网站文件托管：

步骤1 将网站文件上传至OBS的桶中，并设置对象MIME类型。

步骤2 设置对象访问权限为公共读。

步骤3 通过浏览器访问对象。

----结束

代码示例：网站文件托管

以下示例展示如何实现网站文件托管：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何实现网站文件托管：
```



```
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_html_test.html";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.html";
    // 设置MIME类型
    put_properties.content_type = "text/html";
    // 设置对象访问权限为公共读
    put_properties.canned_acl = OBS_CANNED_ACL_PUBLIC_READ;
    put_properties.content_disposition_filename = "";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
```



```
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
```



```
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

15.2 设置桶的网站配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS允许在桶内保存静态的网页资源，如.html网页文件、flash文件、音视频文件等，当客户端通过桶的Website接入点访问这些对象资源时，浏览器可以直接解析出这些支持的网页资源，呈现给最终用户。典型的应用场景有：

- 重定向所有的请求到另外一个站点。
- 设定特定的重定向规则来重定向特定的请求。

调用设置桶的网站配置接口，您可以为指定桶设置网站配置信息。

接口约束

- 尽量避免目标桶名中带有“.”，否则通过HTTPS访问时可能出现客户端校验证书出错。
- 设置桶的网站配置请求消息体的上限是10KB。
- 您必须是桶拥有者或拥有设置桶的网站配置的权限，才能设置桶的网站配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketWebsite权限，如果使用桶策略则需授予PutBucketWebsite

权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。

- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_website_configuration(const obs_options *options,
    obs_set_bucket_redirect_all_conf *set_bucket_redirect_all,
    obs_set_bucket_website_conf *set_bucket_website_conf,
    obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 15-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
set_bucket_redirect_all	**obs_set_bucket_redirect_all_conf** *	必选	**参数解释：** 描述重定向的配置。 **约束限制：** 无
set_bucket_redirect_all->host_name	const char *	必选	**参数解释：** 描述重定向的站点名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
set_bucket_redirect_all->protocol	const char *	可选	**参数解释：** 描述重定向请求时使用的协议。 **约束限制：** 无 **取值范围：** • http协议 • https协议 **默认取值：** http协议
set_bucket_website_conf	obs_set_bucket_website_conf *	必选	**参数解释：** 描述重定向规则website元素的配置。 **约束限制：** 无
set_bucket_website_conf->suffix	const char *	必选	**参数解释：** suffix元素被追加在对文件夹的请求的末尾，例如：suffix配置的是“index.html”，请求的是“samplebucket/images/”，返回的数据将是“samplebucket”桶内名为“images/index.html”的对象的内容。 **约束限制：** suffix元素不能为空或者包含“/”字符。 **取值范围：** 无 **默认取值：** 无
set_bucket_website_conf->key	const char *	可选	**参数解释：** 当4XX错误出现时使用的对象的名称。这个元素指定了当错误出现时返回的页面。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
set_bucket_website_conf->routingrule_info	bucket_website_routingrule *	可选	**参数解释：** 重定向规则的具体描述。 **约束限制：** 无
set_bucket_website_conf->routingrule_count	int	可选	**参数解释：** set_bucket_website_conf.routingrule_info的总大小。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	obs_response_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 15-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 15-4 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 15-5 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 15-6 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 15-7 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 15-8 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 15-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-11 obs_set_bucket_redirect_all_conf

参数名称	参数类型	是否必选	描述
host_name	const char *	必选	**参数解释** 描述重定向的站点名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
protocol	const char *	必选	**参数解释** 描述重定向请求时使用的协议。 **约束限制：** 无 **取值范围：** • http协议 • https协议 **默认取值：** http协议

表 15-12 obs_set_bucket_website_conf

参数名称	参数类型	是否必选	描述
suffix	const char *	必选	**参数解释** suffix元素被追加在对文件夹的请求的末尾。 **约束限制：** suffix元素不能为空或者包含“/”字符。 **取值范围：** 无 **默认取值：** 无
key	constchar*	可选	**参数解释** 当4XX错误出现时使用的对象的名称。这个元素指定了当错误出现时返回的页面。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
routingrule_info	**bucket_website_routingrule** *	可选	**参数解释** 重定向规则的元素。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
routingrule_count	int	必选	**参数解释** routingrule_info的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-13 bucket_website_routingrule

参数名称	参数类型	是否必选	描述
key_prefix_equals	const char *	必选	**参数解释** 描述当重定向生效时对象名的前缀。 例如：重定向ExamplePage.html对象的请求，key_prefix_equals设为ExamplePage.html。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
http_errorcode_returned_equals	const char *	必选	**参数解释** 描述Redirect生效时的HTTP错误码。当发生错误时，如果错误码等于这个值，那么Redirect生效。 例如： 当返回的http错误码为404时重定向到NotFound.html，可以将http_errorcode_returned_equals设置为404，replace_key_with设置为NotFound.html。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
protocol	const char *	必选	**参数解释** 描述重定向请求时使用的协议（http，https）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
host_name	const char *	必选	**参数解释** 描述重定向的站点名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
replace_key_prefix_with	const char *	必选	**参数解释** 描述重定向请求时使用的对象名前缀，请求中的对象名会将key_prefix_equals的内容替换为replace_key_prefix_with的内容。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
replace_key_with	const char *	必选	**参数解释** 描述重定向请求时使用的对象名，请求中的整个对象名会被替换为replace_key_with的内容。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
http_redirect_code	const char *	必选	**参数解释** 描述响应中的HTTP状态码。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-14 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 15-15 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-16 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-17 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释:** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
request_id2	const char *	可选	**参数解释:** 帮助定位问题的特殊符号。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_type	const char *	可选	**参数解释:** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无
content_length	uint64_t	可选	**参数解释:** 响应消息体的字节长度。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

参数名称	参数类型	是否必选	描述
server	const char *	可选	**参数解释：** 请求中的Server头域 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-18 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value *	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-19 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-20 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例一：配置默认主页错误页面和重定向规则

以下示例展示如何配置默认主页错误页面和重定向规则：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何设置桶的托管配置：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    obs_set_bucket_website_conf set_bucket_website_conf;
    // 配置默认主页
    set_bucket_website_conf.suffix = "index.html";
    // 配置错误页面
```

```
    set_bucket_website_conf.key = "Error.html";
    // 定义重定向规则
    set_bucket_website_conf.routingrule_count = 2;
    bucket_website_routingrule temp[2];
    memset(&temp[0], 0, sizeof(bucket_website_routingrule));
    memset(&temp[1], 0, sizeof(bucket_website_routingrule));
    set_bucket_website_conf.routingrule_info = temp;
    temp[0].key_prefix_equals = "key_prefix1";
    temp[0].replace_key_prefix_with = "replace_key_prefix1";
    temp[0].http_errorcode_returned_equals = "404";
    temp[0].http_redirect_code = NULL;
    temp[0].host_name = "www.example.com";
    temp[0].protocol = "http";
    temp[1].key_prefix_equals = "key_prefix2";
    temp[1].replace_key_prefix_with = "replace_key_prefix2";
    temp[1].http_errorcode_returned_equals = "404";
    temp[1].http_redirect_code = NULL;
    temp[1].host_name = "www.example.com";
    temp[1].protocol = "http";
    // 设置重定向规则
    set_bucket_website_configuration(&options, NULL, &set_bucket_website_conf,
        &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("create bucket %s successfully. \n", bucketName);
    }
    else {
        printf("create bucket %s failed(%s).\n", bucketName, obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
```



```
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

代码示例二：配置所有请求重定向

以下示例展示如何配置所有请求重定向：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
```



```
int main()
{
    // 以下示例展示如何配置所有请求重定向：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 设置所有请求重定向
    obs_set_bucket_redirect_all_conf set_bucket_redirect_all;
    set_bucket_redirect_all.host_name = "www.example.com";
    set_bucket_redirect_all.protocol = "https";
    set_bucket_website_configuration(&options, &set_bucket_redirect_all, NULL,
        &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket redirect all successfully. \n");
    }
    else
    {
        printf("set bucket redirect all failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                  \
        if (properties-> field) {                             \
            printf("%s: %s\n", name, properties->field);          \
        }                                                 \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
```



```
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于设置桶的网站配置的API说明，请参见**设置桶的网站配置**。

- 更多关于设置桶的Website配置的代码示例，请参见**Github示例**。
- 设置桶的Website配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 静态网站托管相关常见问题请参见**静态网站托管相关常见问题**。

15.3 获取桶的网站配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS允许在桶内保存静态的网页资源，如.html网页文件、flash文件、音视频文件等，当客户端通过桶的Website接入点访问这些对象资源时，浏览器可以直接解析出这些支持的网页资源，呈现给最终用户。典型的应用场景有：

- 重定向所有的请求到另外一个站点。
- 设定特定的重定向规则来重定向特定的请求。

调用获取桶的网站配置接口，您可以获取指定桶的网站配置信息。

接口约束

- 您必须是桶拥有者或拥有获取桶的网站配置的权限，才能获取桶的网站配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketWebsite权限，如果使用桶策略则需授予GetBucketWebsite权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_website_configuration(const obs_options *options,
    obs_get_bucket_websiteconf_handler *handler, void *callback_data);
```

请求参数说明

表 15-21 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过obs_options设置AK、SK、endpoint、bucket、超时时间、临时鉴权。

参数名称	参数类型	是否必选	描述
handler	obs_get_bucket_websiteconf_handler *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-22 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	obs_bucket_context	必选	**参数解释：** 桶相关设置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
request_options	obs_http_request_option	必选	**参数解释：** 请求相关设置。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-23 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 15-24 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 15-25 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-26 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 15-27 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 15-28 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。

枚举值	说明
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 15-29 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-30 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-31 obs_get_bucket_websiteconf_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
get_bucket_website_conf_callback	**obs_get_bucket_websiteconf_callback** *	必选	**参数解释：** 回调函数指针，可以在这个回调中把回调的参数记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 15-32 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 15-33 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-34 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-35 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-36 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-37 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-38 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 15-39 obs_get_bucket_websiteconf_callback

参数名称	参数类型	是否必选	描述
hostname	const char *	必选	**参数解释：** 描述重定向的站点名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
protocol	const char *	必选	**参数解释：** 描述重定向请求时使用的协议（http，https），默认使用http协议。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
suffix	const char *	必选	**参数解释：** suffix元素被追加在对文件夹的请求的末尾，例如：suffix配置的是“index.html”，请求的是“samplebucket/images/”，返回的数据将是“samplebucket”桶内名为“images/index.html”的对象的内容。 **约束限制：** suffix元素不能为空或者包含“/”字符。 **取值范围：** 无 **默认取值：** 无
key	const char *	必选	**参数解释：** 当4XX错误出现时使用的对象的名称。这个元素指定了当错误出现时返回的页面。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
routingrule	const **bucket_website_routingrule** *	必选	**参数解释：** 重定向规则的具体描述。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-40 bucket_website_routingrule

参数名称	参数类型	是否必选	描述
key_prefix_equals	const char *	必选	**参数解释** 描述当重定向生效时对象名的前缀。 例如：重定向ExamplePage.html对象的请求，key_prefix_equals设为ExamplePage.html。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
http_errorcode_returned_equals	const char *	必选	**参数解释** 描述Redirect生效时的HTTP错误码。当发生错误时，如果错误码等于这个值，那么Redirect生效。 例如： 当返回的http错误码为404时重定向到NotFound.html，可以将http_errorcode_returned_equals设置为404，replace_key_with设置为NotFound.html。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
protocol	const char *	必选	**参数解释** 描述重定向请求时使用的协议（http，https）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
host_name	const char *	必选	**参数解释** 描述重定向的站点名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
replace_key_prefix_with	const char *	必选	**参数解释** 描述重定向请求时使用的对象名前缀，请求中的对象名会将key_prefix_equals的内容替换为replace_key_prefix_with的内容。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
replace_key_with	const char *	必选	**参数解释** 描述重定向请求时使用的对象名，请求中的整个对象名会被替换为replace_key_with的内容。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
http_redirect_code	const char *	必选	**参数解释** 描述响应中的HTTP状态码。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：获取桶的网站配置

以下示例展示如何获取桶的网站配置：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status get_bucket_websiteconf_callback(const char *hostname,
    const char *protocol,
    const char *suffix,
    const char *key,
    const bucket_website_routingrule *websiteconf,
    int webdatacount,
    void *callback_data);
int main()
{
    // 以下示例展示如何查看托管配置：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（ access_key_id和
acces_key_secret ）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 设置响应回调函数
    obs_get_bucket_websiteconf_handler get_bucket_websiteconf_handler =
    {
        response_handler,
        &get_bucket_websiteconf_callback
    };
    // 获取桶的托管配置
    get_bucket_website_configuration(&options, &get_bucket_websiteconf_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("get bucket website successfully.\n");
    }
    else
    {
        printf("get bucket website failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
```



```
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
```



```
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
obs_status get_bucket_websiteconf_callback(const char *hostname,
    const char *protocol,
    const char *suffix,
    const char *key,
    const bucket_website_routingrule *websiteconf,
    int webdatacount,
    void *callback_data)
{
    (void)callback_data;
    int i = 0;
    printf("redirectAllRequestsTo hostname : %s\n", hostname);
    printf("redirectAllRequestsTo protocol : %s\n", protocol);
    printf("suffix : %s\n", suffix);
    printf("key : %s\n", key);
    for (i = 0; i < webdatacount; i++)
    {
        printf("key_prefix_equals : %s\n", websiteconf[i].key_prefix_equals);
        printf("http_errorcode_returned_equals : %s\n", websiteconf[i].http_errorcode_returned_equals);
        printf("replace_key_prefix_with : %s\n", websiteconf[i].replace_key_prefix_with);
        printf("replace_key_with : %s\n", websiteconf[i].replace_key_with);
        printf("http_redirect_code : %s\n", websiteconf[i].http_redirect_code);
        printf("hostname : %s\n", websiteconf[i].host_name);
        printf("protocol : %s\n", websiteconf[i].protocol);
    }
    return OBS_STATUS_OK;
}
```

相关链接

- 关于获取桶的网站配置的API说明，请参见**获取桶的网站配置**。
- 更多关于获取桶的网站配置的代码示例，请参见**Github示例**。
- 获取桶的网站配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 静态网站托管相关常见问题请参见**静态网站托管相关常见问题**。

15.4 删除桶的网站配置(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

OBS允许在桶内保存静态的网页资源，如.html网页文件、flash文件、音视频文件等，当客户端通过桶的Website接入点访问这些对象资源时，浏览器可以直接解析出这些支持的网页资源，呈现给最终用户。典型的应用场景有：

- 重定向所有的请求到另外一个站点。
- 设定特定的重定向规则来重定向特定的请求。

调用删除桶的网站配置接口，您可以删除指定桶的网站配置。

接口约束

- 您必须是桶拥有者或拥有删除桶的网站配置的权限，才能删除桶的网站配置。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:DeleteBucketWebsite权限，如果使用桶策略则需授予DeleteBucketWebsite权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void delete_bucket_website_configuration(const obs_options *options,
        obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 15-41 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过**obs_options**设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-42 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 15-43 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 15-44 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 15-45 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 15-46 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。

枚举值	说明
OBS_BUCKET_PFS	并行文件系统。

表 15-47 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 15-48 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）

参数名称	参数类型	是否必选	描述
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-49 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_callback	void(*temp_auth_callback)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-50 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-51 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 15-52 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-53 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 15-54 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-55 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-56 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 15-57 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：删除桶的网站配置

以下示例展示如何删除桶的网站配置：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
int main()
{
    // 以下示例展示如何清除托管配置：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";

    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 删除桶的托管配置
    delete_bucket_website_configuration(&options, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
```



```
        printf("delete bucket website successfully.\n");
    }
    else
    {
        printf("delete bucket website failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
```



```
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于删除桶的网站配置的API说明，请参见**删除桶的网站配置**。
- 更多关于删除桶的网站配置的代码示例，请参见**Github示例**。
- 删除桶的网站配置过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。
- 静态网站托管相关常见问题请参见**静态网站托管相关常见问题**。

16 标签管理(C SDK)

16.1 设置桶标签(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

为桶添加标签后，该桶上所有请求产生的计费话单里都会带上这些标签，从而可以针对话单报表做分类筛选，进行更详细的成本分析。例如：某个应用程序在运行过程会往桶里上传数据，可以用应用名称作为标签，设置到被使用的桶上。在分析话单时，就可以通过应用名的标签来分析此应用的成本。

调用设置桶标签接口，您可为指定桶添加标签。

接口约束

- 每个桶最多能设置10个标签。
- 标签的键名（Key）的最大长度为36个字符，标签的键值（Value）的最大长度为43个字符。
- 标签的键名（Key）和键值（Value）不能包含字符“,”、“*”、“|”、“/”、“<”、“>”、“=”、“\”以及ASCII码0x00--0x1F这些控制字符。
- 您必须是桶拥有者或拥有设置桶标签的权限，才能设置桶标签。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:PutBucketTagging权限，如果使用桶策略则需授予PutBucketTagging权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void set_bucket_tagging(const obs_options *options,obs_name_value * tagging_list,
    unsigned int number, obs_response_handler *handler, void *callback_data);
```

请求参数说明

表 16-1 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过**obs_options**设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
tagging_list	**obs_name_value** *	必选	**参数解释：** 标签列表。 **约束限制：** 无
number	unsigned int	必选	**参数解释：** 标签个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-2 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 16-3 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 16-4 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 16-5 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 16-6 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 16-7 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 16-8 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-9 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释:** 临时鉴权的有效期（单位：秒）。 **约束限制:** 无 **取值范围:** [0-630720000] **默认取值**: 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释:** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 16-10 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-11 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	**obs_response_properties_callback** *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	**obs_response_complete_callback** *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把**obs_status**和**obs_error_details**的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 16-12 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释:** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无
callback_data	void *	必选	**参数解释:** 用户自定义回调数据指针。 **约束限制:** 无 **取值范围:** 无 **默认取值**: 无

表 16-13 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释:** SDK内部的请求状态码。 **约束限制:** 无 **取值范围:** 请详见**obs_status**。
error_details	const **obs_error_details** *	必选	**参数解释:** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制:** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-14 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-15 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶或对象资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-16 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-17 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：设置桶标签

以下示例展示如何设置桶标签：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
#define TAG_LIST_LENGTH 10
int main()
{
    // 以下示例展示如何通过set_bucket_tagging设置桶标签：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 定义桶标签
    char tagKey[][OBS_COMMON_LEN_256] = { {"k1"},{"k2"},{"k3"},{"k4"},{"k5"},{"k6"},{"k7"},{"k8"},{"k9"},{"k10"} };
```



```
    char tagValue[][OBS_COMMON_LEN_256] = { {"v1"},{"v2"},{"v3"},{"v4"},{"v5"},{"v6"},{"v7"},{"v8"},{"v9"},
{"v10"} };
    obs_name_value tagginglist[TAG_LIST_LENGTH] = { 0 };
    int i = 0;
    for (; i < TAG_LIST_LENGTH; i++)
    {
        tagginglist[i].name = tagKey[i];
        tagginglist[i].value = tagValue[i];
    }
    // 设置桶标签
    set_bucket_tagging(&options, tagginglist, TAG_LIST_LENGTH, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
        printf("set bucket tagging successfully. \n");
    }
    else
    {
        printf("set bucket tagging failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
```



```
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

说明

- 每个桶支持最多10个标签。
- 标签的name和value支持Unicode。

相关链接

- 关于设置桶标签的API说明，请参见**设置桶标签**。
- 更多关于设置桶标签的代码示例，请参见**Github示例**。
- 设置桶标签过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

16.2 获取桶标签(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

为桶添加标签后，该桶上所有请求产生的计费话单里都会带上这些标签，从而可以针对话单报表做分类筛选，进行更详细的成本分析。例如：某个应用程序在运行过程会往桶里上传数据，可以用应用名称作为标签，设置到被使用的桶上。在分析话单时，就可以通过应用名的标签来分析此应用的成本。

调用获取桶标签接口，您可获取指定桶的标签。

接口约束

- 您必须是桶拥有者或拥有获取桶标签的权限，才能获取桶标签。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:GetBucketTagging权限，如果使用桶策略则需授予GetBucketTagging权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void get_bucket_tagging(const obs_options *options, obs_get_bucket_tagging_handler *handler,
    void *callback_data);
```

请求参数说明

表 16-18 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过**obs_options**设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
handler	**obs_get_bucket_tagging_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	可选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-19 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 16-20 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false
protocol	**obs_protocol**	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见**obs_protocol**。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	obs_bucket_type	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_type。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）

参数名称	参数类型	是否必选	描述
bucket_list_type	obs_bucket_list_type	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见obs_bucket_list_type。

表 16-21 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 16-22 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 16-23 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 16-24 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 16-25 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-26 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-27 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-28 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 16-29 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const obs_response_properties*	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-30 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	obs_status	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见obs_status。

参数名称	参数类型	是否必选	描述
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-31 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-32 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释:** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制:** 无 **取值范围:** 取值范围可参见**错误码**。 **默认取值:** 无
resource	const char*	**参数解释:** 该错误相关的桶或对象资源。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
further_details	const char*	**参数解释:** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无
extra_details_count	int	**参数解释:** 错误响应消息体XML中的其他元素的数量。 **约束限制:** 无 **取值范围:** 无 **默认取值:** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-33 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-34 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

表 16-35 obs_get_bucket_tagging_handler

参数名称	参数类型	是否必选	描述
response_handler	**obs_response_handler** *	必选	**参数解释：** 响应回调函数结构体。 **约束限制：** 无
get_bucket_tagging_callback	**obs_get_bucket_tagging_callback** *	必选	**参数解释：** 标签回调函数指针，可以在这个回调中把tagging_count和tagging_list的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 16-36 obs_get_bucket_tagging_callback

参数名称	参数类型	是否必选	描述
tagging_count	int	必选	**参数解释：** 标签列表中标签的数目。 **约束限制：** 无 **取值范围：** [0，10] **默认取值：** 无
tagging_list	obs_name_value *	必选	**参数解释：** 标签列表。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

代码示例：获取桶标签

以下示例展示如何获取桶标签：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
obs_status get_bucket_tagging_callback(int tagging_count, obs_name_value *tagging_list, void *callback_data);
typedef struct tagkv
{
    char key[250];
    char value[250];
}tagkv;
typedef struct TaggingInfo
{
    int tagCount;
    tagkv taglist[10];
    obs_status ret_status;
}TaggingInfo;
int main()
{
    // 以下示例展示如何通过get_bucket_tagging查看桶标签：
```

```
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback,
&response_complete_callback };
    // 设置响应回调函数
    obs_get_bucket_tagging_handler get_bucket_tagging_handler =
    {
        response_handler,
        &get_bucket_tagging_callback
    };
    // 创建回调数据
    TaggingInfo tagging_info;
    memset(&tagging_info, 0, sizeof(TaggingInfo));
    tagging_info.ret_status = OBS_STATUS_BUTT;
    // 获取桶标签
    get_bucket_tagging(&options, &get_bucket_tagging_handler, &tagging_info);
    if (OBS_STATUS_OK == tagging_info.ret_status) {
        printf("get bucket tagging successfully.\n");
    }
    else
    {
        printf("get bucket tagging failed(%s).\n", obs_get_status_name(tagging_info.ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
void printTagInfo(TaggingInfo* infoToPrint)
{
    int i;
    printf("etag number is %d\n", infoToPrint->tagCount);
    for (i = 0; i < infoToPrint->tagCount; i++)
    {
        printf("key:[%s], value[%s]\n", infoToPrint->taglist[i].key, infoToPrint->taglist[i].value);
    }
}
obs_status get_bucket_tagging_callback(int tagging_count, obs_name_value *tagging_list, void
*callback_data)
{
    int tag_num = 0;
    TaggingInfo * tag_info = (TaggingInfo *)callback_data;
    tag_info->tagCount = tagging_count;
    if (tagging_count > 0)
    {
        for (tag_num = 0; tag_num < tagging_count; tag_num++)
        {
            memcpy_s(tag_info->taglist[tag_num].key, sizeof(tag_info->taglist[tag_num].key),
(&tagging_list[tag_num])->name, strlen((&tagging_list[tag_num])->name) + 1);
            memcpy_s(tag_info->taglist[tag_num].value, sizeof(tag_info->taglist[tag_num].value),
(&tagging_list[tag_num])->value, strlen((&tagging_list[tag_num])->value) + 1);
        }
    }
    printTagInfo(tag_info);
    return OBS_STATUS_OK;
}
```



```
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        TaggingInfo *taggingInfo = (TaggingInfo *)callback_data;
        taggingInfo->ret_status = status;
    }
    else {
```



```
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于获取桶标签的API说明，请参见**获取桶标签**。
- 更多关于获取桶标签的代码示例，请参见**Github示例**。
- 获取桶标签过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

16.3 删除桶标签(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能介绍

为桶添加标签后，该桶上所有请求产生的计费话单里都会带上这些标签，从而可以针对话单报表做分类筛选，进行更详细的成本分析。例如：某个应用程序在运行过程会往桶里上传数据，可以用应用名称作为标签，设置到被使用的桶上。在分析话单时，就可以通过应用名的标签来分析此应用的成本。

调用删除桶标签接口，您可删除指定桶的标签。

接口约束

- 您必须是桶拥有者或拥有删除桶标签的权限，才能删除桶标签。建议使用IAM或桶策略进行授权，如果使用IAM则需授予obs:bucket:DeleteBucketTagging权限，如果使用桶策略则需授予DeleteBucketTagging权限。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**自定义创建桶策略**。

- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

方法定义

```
void delete_bucket_tagging(const obs_options *options, obs_response_handler *handler,
    void *callback_data);
```

请求参数说明

表 16-37 请求参数列表

参数名称	参数类型	是否必选	描述
options	const **obs_options***	必选	**参数解释：** 请求桶的上下文，**配置option(C SDK)**，通过**obs_options**设置AK、SK、endpoint、bucket、超时时间、临时鉴权。 **约束限制：** 无
handler	**obs_response_handler** *	必选	**参数解释：** 回调结构体，结构体内所有成员都是回调函数的指针，用于设置处理接口响应数据的回调函数。您可以通过设置回调函数，把服务端的响应数据复制到您的自定义回调数据callback_data中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-38 obs_options

参数名称	参数类型	是否必选	描述
bucket_options	**obs_bucket_context**	必选	**参数解释：** 桶相关设置。 **约束限制：** 无
request_options	**obs_http_request_option**	必选	**参数解释：** 请求相关设置。 **约束限制：** 无
temp_auth	**temp_auth_configure***	可选	**参数解释：** 用于临时计算签名的结构体，不使用时请设置为NULL。 **约束限制：** 无

表 16-39 obs_bucket_context

参数名称	参数类型	是否必选	描述
host_name	char *	必选	**参数解释：** 连接OBS的服务地址，请求使用的主机名，是指存放资源的服务器的域名，就是终端节点endpoint。 示例：host_name = "obs.cn-north-4.myhuaweicloud.com"; **约束限制：** 不需要带“http://”或“https://”前缀，通过**obs_protocol**控制是使用http协议还是https协议。 **取值范围：** 您可以从**这里**查看OBS当前开通的服务地址。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_name	char *	必选	**参数解释：** 桶名。 **约束限制：** ● 桶的名字需全局唯一，不能与已有的任何桶名称重复，包括其他用户创建的桶。 ● 桶命名规则如下： – 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。 – 禁止使用IP地址。 – 禁止以“-”或“.”开头及结尾。 – 禁止两个“.”相邻（如：“my..bucket”）。 – 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。 ● 同一用户在同一个区域多次创建同名桶不会报错，创建的桶属性以第一次请求为准。 **取值范围：** 无 **默认取值：** 无
useCname	bool	可选	**参数解释：** 是否通过自定义域名访问OBS服务。 **约束限制：** 无 **取值范围：** true：通过自定义域名访问OBS服务。 false：不通过自定义域名访问OBS服务。 **默认取值：** false

参数名称	参数类型	是否必选	描述
protocol	obs_protocol	可选	**参数解释：** 是使用http协议还是https协议。 **约束限制：** 无 **取值范围：** 请详见obs_protocol。 **默认取值：** OBS_PROTOCOL_HTTPS（默认使用https协议）
access_key	char *	必选	**参数解释：** 访问密钥中的AK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
secret_access_key	char *	必选	**参数解释：** 访问密钥中的SK。 **约束限制：** 无 **取值范围：** 请详见创建访问密钥。 **默认取值：** 无
storage_class	obs_storage_class	可选	**参数解释：** 创桶时可指定的桶的存储类别。 **约束限制：** 无 **取值范围：** 请详见obs_storage_class。 **默认取值：** OBS_STORAGE_CLASS_STANDARD（指标准存储类别）

参数名称	参数类型	是否必选	描述
token	char *	可选	**参数解释：** 临时访问密钥中的SecurityToken。 **约束限制：** 无 **取值范围：** 请详见**创建访问密钥**。 **默认取值：** 无
epid	char *	可选	**参数解释：** 创桶时可指定的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** epid格式为uuid，未开通企业项目的用户可以不带该头域。 示例：9892d768-2d13-450f-aac7-ed0e44c2585f **默认取值：** 无
bucket_type	**obs_bucket_type**	可选	**参数解释：** 创桶时，指定是对象桶还是并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_type**。 **默认取值：** OBS_BUCKET_OBJECT（指对象桶）
bucket_list_type	**obs_bucket_list_type**	可选	**参数解释：** 列举桶时，确定列举桶的类型：所有桶、对象桶、并行文件系统。 **约束限制：** 无 **取值范围：** 请详见**obs_bucket_list_type**。

表 16-40 obs_protocol

枚举值	说明
OBS_PROTOCOL_HTTPS	使用https协议访问。
OBS_PROTOCOL_HTTP	使用http协议访问。

表 16-41 obs_storage_class

枚举值	说明
OBS_STORAGE_CLASS_STANDARD	标准存储。 标准存储拥有低访问时延和较高的吞吐量，适用于有大量热点对象（平均一个月多次）或小对象（<1MB），且需要频繁访问数据的业务场景。
OBS_STORAGE_CLASS_STANDARD_IA	低频访问存储。 低频访问存储适用于不频繁访问（平均一年少于12次）但在需要时也要求能够快速访问数据的业务场景。
OBS_STORAGE_CLASS_GLACIER	归档存储。 归档存储适用于很少访问（平均一年访问一次）数据的业务场景。
OBS_STORAGE_CLASS_DEEP_ARCHIVE	深度归档存储（受限公测） 适用于长期不访问（平均几年访问一次）数据的业务场景。

表 16-42 obs_bucket_type

枚举值	说明
OBS_BUCKET_OBJECT	对象桶。
OBS_BUCKET_PFS	并行文件系统。

表 16-43 obs_bucket_list_type

枚举值	说明
OBS_BUCKET_LIST_ALL	列举所有桶。
OBS_BUCKET_LIST_OBJECT	列举所有对象桶。
OBS_BUCKET_LIST_PFS	列举所有并行文件系统。

表 16-44 obs_http_request_option

参数名称	参数类型	是否必选	描述
connect_time	int	必选	**参数解释：** 建立HTTP/HTTPS连接的超时时间（单位：毫秒）。 **约束限制：** 无 **取值范围：** [10000, 60000] **默认取值：** 60000
max_connected_time	int	必选	**参数解释：** 请求超时时间（单位：秒）。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 0（指永远不会主动断开链接）
proxy_auth	char*	可选	**参数解释：** 代理认证信息，格式为username:password。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
proxy_host	char*	可选	**参数解释：** 代理服务器的IP地址或主机名。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-45 temp_auth_configure

参数名称	参数类型	是否必选	描述
expires	long long int	必选	**参数解释：** 临时鉴权的有效期（单位：秒）。 **约束限制：** 无 **取值范围：** [0-630720000] **默认取值**： 无
temp_auth_callback	void(***temp_auth_callback**)(char *temp_auth_url, uint64_t temp_auth_url_len, char*temp_auth_headers, uint64_t temp_auth_headers_len, void*callback_data)	必选	**参数解释：** 用户自定义回调函数指针，用于将临时url以及涉及的计算签名头域记录到用户自定义回调数据中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-46 temp_auth_callback

参数名称	参数类型	是否必选	描述
temp_auth_url	char *	必选	**参数解释：** 临时鉴权的URL。OBS服务支持用户构造一个特定操作的URL，这个URL的Query参数中会包含用户AK、签名、有效期等信息，任何拿到这个URL的人均可执行临时鉴权操作，OBS服务收到这个请求后认为该请求就是签发URL用户自己在执行操作。例如构造一个携带签名信息的下载对象的URL，拿到相应URL的人都能下载这个对象，但该URL只在Expires指定的失效时间内有效。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_url_len	uint64_t	必选	**参数解释：** 临时鉴权的URL的长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
temp_auth_headers	char *	必选	**参数解释：** 临时鉴权的headers。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
temp_auth_headers_len	uint64_t	必选	**参数解释：** 临时鉴权的headers的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-47 obs_response_handler

参数名称	参数类型	是否必选	描述
properties_callback	obs_response_properties_callback *	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
complete_callback	obs_response_complete_callback *	必选	**参数解释：** 结束回调函数指针，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

表 16-48 obs_response_properties_callback

参数名称	参数类型	是否必选	描述
properties	const **obs_response_properties***	必选	**参数解释：** 响应头域中的参数，建议将其内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-49 obs_response_complete_callback

参数名称	参数类型	是否必选	描述
status	**obs_status**	必选	**参数解释：** SDK内部的请求状态码。 **约束限制：** 无 **取值范围：** 请详见**obs_status**。
error_details	const **obs_error_details***	必选	**参数解释：** 响应回调函数指针，可以在这个回调中把properties的内容记录到callback_data（用户自定义回调数据）中。 **约束限制：** 无

参数名称	参数类型	是否必选	描述
callback_data	void *	必选	**参数解释：** 用户自定义回调数据指针。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

表 16-50 obs_response_properties

参数名称	参数类型	是否必选	描述
request_id	const char *	可选	**参数解释：** 由OBS创建来唯一确定本次请求的值，可以通过该值来定位问题。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
request_id2	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
content_type	const char *	可选	**参数解释：** 对象的文件类型（MIME类型）。Content-Type（MIME）用于标识发送或接收数据的类型，浏览器根据该参数来决定数据的打开方式。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
content_length	uint64_t	可选	**参数解释：** 响应消息体的字节长度。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无
server	const char *	可选	**参数解释：** HTTP请求中的Server头域。 **约束限制：** 无 **取值范围：** 无 **默认取值**： 无

参数名称	参数类型	是否必选	描述
etag	const char *	可选	**参数解释：** 对象的base64编码的128位MD5摘要。ETag是对象内容的唯一标识，可以通过该值识别对象内容是否有变化。比如上传对象时ETag为A，下载对象时ETag为B，则说明对象内容发生了变化。ETag只反映变化的内容，而不是其元数据。上传的对象或复制操作创建的对象，都有唯一的ETag。 **约束限制：** 当对象是服务端加密的对象时，ETag值不是对象的MD5值。 **取值范围：** 长度为32的字符串。 **默认取值：** 无
expiration	const char *	可选	**参数解释：** 对象的详细过期信息。 **约束限制：** 无 **取值范围：** 大于0的整型数，单位：天。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
website_redirect_location	const char *	可选	**参数解释：** 当桶设置了Website配置，就可以设置对象元数据的这个属性，Website接入点返回301重定向响应，将获取这个对象的请求重定向到该属性指定的桶内另一个对象或一个外部的URL，该参数指明对象的重定向地址。 例如，重定向请求到桶内另一对象： x-obs-website-redirect-location:/anotherPage.html 或重定向请求到一个外部URL： x-obs-website-redirect-location:http://www.example.com/ OBS将这个值从头域中取出，保存在对象的元数据“WebsiteRedirectLocation”中。 **约束限制：** ● 必须以“/”、“http://”或“https://”开头，长度不超过2KB。 ● OBS仅支持为桶根目录下的对象设置重定向，不支持为桶中文件夹下的对象设置重定向。 **取值范围：** 无 **默认取值：** 无
version_id	const char *	可选	**参数解释：** 对象的版本号。 **约束限制：** 如果该对象无版本号，则为NULL。 **取值范围：** 长度为32的字符串。 **默认取值**： 无

参数名称	参数类型	是否必选	描述
meta_data_count	int	可选	**参数解释：** meta_data数组中的元素个数。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
meta_data	const **obs_name_value** *	可选	**参数解释：** 对象的自定义元数据。OBS支持用户使用以“x-obs-meta-”开头的消息头来加入自定义的元数据，以便对对象进行自定义管理。当用户获取此对象或查询此对象元数据时，加入的自定义元数据将会在返回的消息头中出现。 **约束限制：** 无
use_server_side_encryption	char	可选	**参数解释：** 如果开启了服务端加密，会被置为'\1'。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
allow_origin	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求中的Origin满足服务端的CORS配置，则在响应中包含这个Origin。 **约束限制：** 无 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
allow_headers	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的headers满足服务端的CORS配置，则在响应中包含这个headers。 **约束限制：** 最多可填写一个“*”通配符，不支持&、:、<、空格以及中文字符。 **取值范围：** 符合CORS协议的取值范围。 **默认取值：** 无
max_age	const char *	可选	**参数解释：** 桶CORS规则中的MaxAgeSeconds。MaxAgeSeconds指请求来源的客户端可以对跨域请求返回结果的缓存时间。 **约束限制：** 每个CORSRule可以包含至多一个MaxAgeSeconds。 **取值范围：** 大于等于0的整型数，单位：秒。 **默认取值：** 3000
allow_methods	const char *	可选	**参数解释：** 当桶设置了CORS配置，如果请求的Access-Control-Request-Method满足服务端的CORS配置，则在响应中包含这条rule中的Methods。 **约束限制：** 无 **取值范围：** ● GET ● PUT ● HEAD ● POST ● DELETE **取值范围：** 无

参数名称	参数类型	是否必选	描述
expose_headers	const char *	可选	**参数解释：** 桶CORS规则中的ExposeHeader。ExposeHeader是指CORS规则允许响应中可返回的附加头域，给客户端提供额外的信息。默认情况下浏览器只能访问以下头域：Content-Length、Content-Type，如果需要访问其他头域，需要在附加头域中配置。 **约束限制：** 不支持*、&、:、<、空格以及中文字符。 **取值范围：** 无 **默认取值：** 无
storage_class	const char *	可选	**参数解释：** 对象的存储类别。 **约束限制：** 仅当对象为非标准存储类别时，会返回此头域。 **取值范围：** ● WARM（指低频存储） ● COLD（指归档存储） ● DEEP_ARCHIVE（指深度归档存储） **默认取值：** 无
server_side_encryption	const char *	可选	**参数解释：** 该头域表示服务端的加密方式。 示例：x-obs-server-side-encryption：kms **约束限制：** 如果服务端加密是SSE-KMS方式，响应包含该头域。 **取值范围：** ● kms（指SSE-KMS加密方式） ● obs（指SSE-OBS加密方式） **默认取值：** 无

参数名称	参数类型	是否必选	描述
kms_key_id	const char *	可选	**参数描述：** 密钥ID。当加密方式为SSE-KMS且使用指定密钥加密时，需输入密钥ID。 **约束限制：** 当您设置了server_side_encryption且赋值为“kms”，即选择kms加密方式时，才能使用该头域指定加密密钥。 **取值范围：** 无 **默认取值：** 当您选择使用kms加密方式，但未设置此头域时，默认的主密钥将会被使用。如果默认主密钥不存在，系统将默认创建并使用。
customer_algorithm	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的算法。 **约束限制：** 无 **取值范围：** AES256（指AES256解密算法） **默认取值：** 无
customer_key_md5	const char *	可选	**参数解释：** 如果服务端加密是SSE-C方式，响应包含该头域，该头域表示解密使用的密钥的MD5值。 **约束限制：** 由密钥值经过MD5加密再经过Base64编码后得到，示例：4XvB3tbNTN+tIEVa0/fGaQ==。 **取值范围：** 密钥ID MD5的base64值。 **默认取值：** 无

参数名称	参数类型	是否必选	描述
bucket_location	const char *	可选	**参数解释：** 桶的区域位置信息。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
obs_version	const char *	可选	**参数解释：** 桶所在的OBS服务版本号。 **约束限制：** 无 **取值范围：** ● 3.0：最新版本的桶。 ● --：表示老版本的桶。 **默认取值：** 无
restore	const char *	可选	**参数解释：** 标识对象的恢复状态。 示例：正在恢复ongoing-request="true"；已恢复 ongoing-request="false", expiry-date="Wed, 7 Nov 2012 00:00:00 GMT"。其中expiry-date表示对象恢复后的失效时间。 **约束限制：** 仅当对象为归档或深度归档存储类别，并且处于正在恢复或已经恢复时，会返回此头域。 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
obs_object_type	const char *	可选	**参数解释：** 对象的类型。 **约束限制：** 仅当对象为非Normal对象时，会返回此头域。 **取值范围：** Appendable **默认取值：** 无
obs_next_append_position	const char *	可选	**参数解释：** 指明下一次请求应该提供的position。 **约束限制：** 仅当对象为Appendable对象时，会返回此头域。 **取值范围：** 无 **默认取值：** 无
obs_head_epid	const char *	可选	**参数解释：** 当前桶的企业项目ID，开通企业项目的用户可以从企业项目服务获取。 **约束限制：** 格式为uuid，未开通企业项目的用户可以不带该头域。 **取值范围：** 无 **默认取值：** 无
reserved_indicator	const char *	可选	**参数解释：** 帮助定位问题的特殊符号。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-51 obs_error_details

参数名称	参数类型	描述
message	const char*	**参数解释：** 错误响应消息体XML中具体错误更全面、详细的英文解释。 **约束限制：** 无 **取值范围：** 取值范围可参见**错误码**。 **默认取值：** 无
resource	const char*	**参数解释：** 该错误相关的桶资源。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
further_details	const char*	**参数解释：** 错误响应消息体XML中的FurtherDetails元素的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
extra_details_count	int	**参数解释：** 错误响应消息体XML中的其他元素的数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	描述
extra_details	obs_name_value*	**参数解释：** 错误响应消息体XML中的其他元素的值。 **约束限制：** 无
error_headers_count	int	**参数解释：** error_headers的头域数量。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无
error_headers	char**	**参数解释：** 响应头域中包含error的所有头域。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-52 obs_name_value

参数名称	参数类型	是否必选	描述
name	char *	可选	**参数解释：** 属性的键。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

参数名称	参数类型	是否必选	描述
value	char *	可选	**参数解释：** 属性的值。 **约束限制：** 无 **取值范围：** 无 **默认取值：** 无

表 16-53 obs_status

枚举值	说明
OBS_STATUS_OK	请求成功。
OBS_STATUS_InitCurlFailed	初始化curl失败。
OBS_STATUS_InternalError	内部错误。
OBS_STATUS_OutOfMemory	本地环境内存不足。
OBS_STATUS_FailedToIInitializeRequest	初始化请求失败。
OBS_STATUS_ConnectionFailed	网络连接失败。
OBS_STATUS_XmlParseFailure	xml解析失败。
OBS_STATUS_NameLookupError	域名解析失败。
OBS_STATUS_FailedToConnect	无法连接到服务端。
OBS_STATUS_PartialFile	网络传输。
OBS_STATUS_InvalidParameter	参数非法。
OBS_STATUS_NoToken	当前并发数已经超过最大并发数（默认值1000），通过set_online_request_max_count函数去调整最大并发数。

枚举值	说明
OBS_STATUS_OpenFileFailed	打开文件失败。
OBS_STATUS_AccessDenied	请求被拒绝。
OBS_STATUS_MalformedPolicy	请求policy格式不正确。
OBS_STATUS_MalformedXML	请求xml格式不正确。
OBS_STATUS_MethodNotAllowed	请求方法不允许。
OBS_STATUS_SignatureDoesNotMatch	签名不匹配，检查ak、sk、token是否对应或有误。
OBS_STATUS_ServiceUnavailable	服务端异常。
OBS_STATUS_SlowDown	请求频率过快。

代码示例：删除桶标签

以下示例展示如何删除桶标签：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
#define TAG_LIST_LENGTH 10
int main()
{
    // 以下示例展示如何通过delete_bucket_tagging删除桶标签：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    obs_response_handler response_handler = { &response_properties_callback, &response_complete_callback };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 删除桶标签
    delete_bucket_tagging(&options, &response_handler, &ret_status);
    if (OBS_STATUS_OK == ret_status) {
```



```
        printf("delete bucket tagging successfully.\n");
    }
    else
    {
        printf("delete bucket tagging failed(%s).\n", obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                        \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
```



```
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n    %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n    %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n    %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n     %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
```

相关链接

- 关于删除桶标签的API说明，请参见**删除桶标签**。
- 更多关于删除桶标签的代码示例，请参见**Github示例**。
- 删除桶标签过程中返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

17 其他接口(C SDK)

17.1 服务端加密(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

用户可以使用普通方式上传、下载对象，也可以使用服务端加密方式进行上传、下载对象。

OBS支持服务端加密功能，使加密的行为在服务端进行。

用户可以根据自身的需求，使用不同的密钥管理方式来使用服务端加密功能。当前支持以下两种都采用行业标准的AES256加密算法的服务端加密方式。

- SSE-KMS方式，OBS使用KMS（Key Management Service）服务提供的密钥进行服务端加密。
- SSE-C方式：客户提供加密密钥的服务端加密方式，即OBS使用用户提供的密钥和密钥的MD5值进行服务端加密。

使用服务端加密，返回的ETag值不是对象的MD5值。无论是否使用服务端加密上传对象，请求消息头中加入Content-MD5参数时，OBS均会对对象进行MD5校验。

更多关于服务端加密的内容请参考**服务端加密**。

接口约束

- 您必须具有执行PutEncryptionConfiguration操作的权限，才能进行服务端加密操作。桶拥有者默认具有此权限，并且可以将此权限授予其他人。相关授权方式介绍可参见**OBS权限控制概述**，配置方式详见**使用IAM自定义策略**、**配置对象策略**。
- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。

支持接口

OBS C SDK支持服务端加密的接口见下表：

OBS C SDK接口方法	描述	支持加密类型
put_object	上传对象时设置加密算法、密钥，对对象启用服务端加密。	SSE-KMS SSE-C
get_object	下载对象时设置解密算法、密钥，用于解密对象。	SSE-C
copy_object	1. 复制对象时设置源对象的解密算法、密钥，用于解密源对象。 2. 复制对象时设置目标对象的加密算法、密钥，对目标对象启用加密算法。	SSE-KMS SSE-C
get_object_metadata	获取对象元数据时设置解密算法、密钥，用于解密对象。	SSE-C
initiate_multi_part_upload	初始化分段上传任务时设置加密算法、密钥，对分段上传任务最终生成的对象启用服务端加密。	SSE-KMS SSE-C
upload_part	上传段时设置加密算法、密钥，对分段数据启用服务端加密。	SSE-C
copy_part	1. 复制段时设置源对象的解密算法、密钥，用于解密源对象。 2. 复制段时设置目标段的加密算法、密钥，对目标段启用加密算法。	SSE-C

代码示例一：上传对象加密

以下示例展示如何在上传对象时使用服务端加密功能：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何在上传对象时使用服务端加密功能：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
```

```
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 使用服务端加密功能时，必须使用https协议
    options.bucket_options.protocol = OBS_PROTOCOL_HTTPS;
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_put_file_test";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    //服务端加密 SSE加密
    server_side_encryption_params encryption_params;
    memset(&encryption_params, 0, sizeof(server_side_encryption_params));
    encryption_params.ssec_customer_algorithm = "AES256";
    encryption_params.ssec_customer_key =
        "K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=";
    put_object(&options, key, content_length, &put_properties, &encryption_params, &putobjectHandler,
&data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
```



```
            data->buffer, data->buffer_len);
        return OBS_STATUS_OK;
    }
    else {
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
print_nonnull("request_id", request_id);
print_nonnull("request_id2", request_id2);
print_nonnull("content_type", content_type);
if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
}
print_nonnull("server", server);
print_nonnull("ETag", etag);
print_nonnull("expiration", expiration);
print_nonnull("website_redirect_location", website_redirect_location);
print_nonnull("version_id", version_id);
print_nonnull("allow_origin", allow_origin);
print_nonnull("allow_headers", allow_headers);
print_nonnull("max_age", max_age);
print_nonnull("allow_methods", allow_methods);
print_nonnull("expose_headers", expose_headers);
print_nonnull("storage_class", storage_class);
print_nonnull("server_side_encryption", server_side_encryption);
print_nonnull("kms_key_id", kms_key_id);
print_nonnull("customer_algorithm", customer_algorithm);
print_nonnull("customer_key_md5", customer_key_md5);
print_nonnull("bucket_location", bucket_location);
print_nonnull("obs_version", obs_version);
print_nonnull("restore", restore);
print_nonnull("obs_object_type", obs_object_type);
print_nonnull("obs_next_append_position", obs_next_append_position);
print_nonnull("obs_head_epid", obs_head_epid);
print_nonnull("reserved_indicator", reserved_indicator);
int i;
for (i = 0; i < properties->meta_data_count; i++) {
    printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
        properties->meta_data[i].value);
}
return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
```



```
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

代码示例二：下载对象解密

以下示例展示如何在下载对象时使用服务端解密功能：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何在下载对象时使用服务端解密功能：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
```



```
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    // 使用服务端加密功能时，必须使用https协议
    options.bucket_options.protocol = OBS_PROTOCOL_HTTPS;
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    //服务端加密 SSEC加密
    server_side_encryption_params encryption_params;
    memset(&encryption_params, 0, sizeof(server_side_encryption_params));
    encryption_params.ssec_customer_algorithm = "AES256";
    encryption_params.ssec_customer_key =
        "K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=";
    get_object(&options, &object_info, &getcondition, &encryption_params, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
// 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                           \
            printf("%s: %s\n", name, properties->field);        \
        }                                                   \
```

```
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
```



```
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```

相关链接

- 关于服务端加密的API说明，请参见**服务端加密简介**。
- 更多关于服务端加密的示例代码，请参见**Github示例**。
- 服务端加密接口返回的错误码含义、问题原因及处理措施可参考**OBS错误码**。

17.2 使用临时 URL 进行授权访问(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

功能说明

临时授权访问是指通过访问密钥、请求方法类型、请求参数等信息生成一个临时访问权限的URL，这个URL中会包含鉴权信息，您可以使用该URL进行访问OBS服务进行特定操作。在生成URL时，您需要指定URL的有效期。生成临时授权访问的URL是通过设置结构体temp_auth_configure来实现的。

temp_auth_configure结构体存在于obs_options结构体中。该方法适用于每个C SDK接口。

参数	作用	SDK中对应的结构体
expires	生成的临时URL的有效期	obs_options. temp_auth_configure
temp_auth_callback	回调函数用于返回生成的临时URL	
callback_data	回调数据	

接口约束

- OBS支持的Region与Endpoint的对应关系，详细信息请参见**地区与终端节点**。
- 如果遇到跨域报错、签名不匹配问题，请参考以下步骤排查问题：
  a. 未配置跨域，需要在控制台配置CORS规则，请参考**配置桶允许跨域请求**。
  b. 签名计算问题，请参考**URL中携带签名**排查签名参数是否正确；比如上传对象功能，后端将Content-Type参与计算签名生成授权URL，但是前端使用授权URL时没有设置Content-Type字段或者传入错误的值，此时会出现跨域错误。解决方案为：Content-Type字段前后端保持一致。

方法定义

```
void (*temp_auth_callback)(char *temp_auth_url, uint64_t temp_auth_url_len, char *temp_auth_headers,
    uint64_t temp_auth_headers_len, void *callback_data);
```

通过OBS C SDK生成临时URL访问OBS的步骤如下：

步骤1 按照下面的代码示例，调用任意SDK接口生成带签名信息的URL和header。

步骤2 使用任意HTTP库发送HTTP/HTTPS请求，访问OBS服务。

----结束

以下代码示例展示了如何使用临时URL进行授权访问，包括：创建桶、上传对象、下载对象、列举对象、删除对象。

代码示例一：生成用于创建桶的临时授权 URL

以下示例展示如何生成用于创建桶的临时授权URL：

```
#include "eSDKOBS.h"
#include <stdio.h>
#define MAX_TEMP_URL_LEN 1024
#define MAX_HEADER_LEN 1024
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct __tempAuthResult
{
    char tmpAuthUrl[MAX_TEMP_URL_LEN];
    char actualHeaders[MAX_HEADER_LEN];
}tempAuthResult;
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData);
int main()
{
    // 以下示例展示如何生成创建桶的URL：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    temp_auth_configure tempauth;
    tempAuthResult  ptrResult;
    memset(&ptrResult, 0, sizeof(tempAuthResult));
    //回调数据
    tempauth.callback_data = (void *)(&ptrResult);
    // 有效时间
    tempauth.expires = 10;
    // 回调函数 返回生成的临时URL
    tempauth.temp_auth_callback = &tempAuthCallBack_getResult;
    options.temp_auth = &tempauth;
    // 回调函数赋值
```



```
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    // 接口调用
    create_bucket(&options, OBS_CANNED_ACL_PRIVATE, NULL, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("the temporary signature url of create bucket generated successfully. \n"
            "the temporary signature url is %s. \n"
            "the actualHeaders are %s. \n", ptrResult.tmpAuthUrl, ptrResult.actualHeaders);
    }
    else
    {
        printf(" the temporary signature url of create bucket generation failed(%s).\n",
obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
```



```
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData)
{
    int urlLen = 0;
    tempAuthResult * ptrResult = (tempAuthResult *)callbackData;
    urlLen = strlen(tempAuthUrl);
    strcpy_s(ptrResult->tmpAuthUrl, MAX_TEMP_URL_LEN, tempAuthUrl);
    strcpy_s(ptrResult->actualHeaders, MAX_HEADER_LEN, tempAuthActualHeaders);
}
```

代码示例二：生成用于上传对象的临时授权 URL

以下示例展示如何生成用于上传对象的临时授权URL：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
```



```
typedef struct __tempAuthResult
{
    char tmpAuthUrl[1024];
    char actualHeaders[1024];
}tempAuthResult;
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData);
int main()
{
    // 以下示例展示如何生成上传对象的URL：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    temp_auth_configure tempauth;
    tempAuthResult  ptrResult;
    memset(&ptrResult, 0, sizeof(tempAuthResult));
    //回调数据
    tempauth.callback_data = (void *)(&ptrResult);
    // 有效时间
    tempauth.expires = 10;
    // 回调函数 返回生成的临时URL
    tempauth.temp_auth_callback = &tempAuthCallBack_getResult;
    options.temp_auth = &tempauth;
    // 回调函数赋值
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* key = "example-object-key";
    obs_put_object_handler putobjectHandler =
    {
        response_handler,
        NULL
    };
    // 初始化结构体put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 接口调用
    put_object(&options, key, 0, &put_properties, 0, &putobjectHandler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("the temporary signature url of put object generated successfully. \n"
            "the temporary signature url is %s. \n"
            "the actualHeaders are %s. \n", ptrResult.tmpAuthUrl, ptrResult.actualHeaders);
    }
    else
    {
        printf(" the temporary signature url of put object generation failed(%s).\n",
obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
```



```
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
```



```
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData)
{
    int urlLen = 0;
    tempAuthResult * ptrResult = (tempAuthResult *)callbackData;
    urlLen = strlen(tempAuthUrl);
    strcpy_s(ptrResult->tmpAuthUrl, 1024, tempAuthUrl);
    strcpy_s(ptrResult->actualHeaders, 1024, tempAuthActualHeaders);
}
```

代码示例三：生成用于下载对象的临时授权 URL

以下示例展示如何生成用于下载对象的临时授权URL：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct __tempAuthResult
{
    char tmpAuthUrl[1024];
    char actualHeaders[1024];
}tempAuthResult;
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData);
int main()
{
    // 以下示例展示如何生成下载对象的URL：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
```



```
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    temp_auth_configure tempauth;
    tempAuthResult  ptrResult;
    memset(&ptrResult, 0, sizeof(tempAuthResult));
    //回调数据
    tempauth.callback_data = (void *)(&ptrResult);
    // 有效时间
    tempauth.expires = 10;
    // 回调函数 返回生成的临时URL
    tempauth.temp_auth_callback = &tempAuthCallBack_getResult;
    options.temp_auth = &tempauth;
    // 回调函数赋值
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* key = "example-object-key";
    char* versionid = NULL;
    // 下载对象信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = key;
    object_info.version_id = versionid;
    obs_get_object_handler getobjectHandler =
    {
        response_handler,
        NULL
    };
    // 初始化结构体put_properties
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 接口调用
    get_object(&options, &object_info, 0, 0, &getobjectHandler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("the temporary signature url of get object generated successfully. \n"
            "the temporary signature url is %s. \n"
            "the actualHeaders are %s. \n", ptrResult.tmpAuthUrl, ptrResult.actualHeaders);
    }
    else
    {
        printf(" the temporary signature url of get object generation failed(%s).\n",
obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
```



```
        printf("error! obs_sever_callback_data is null!");
        return OBS_STATUS_OK;
    }
}
// 打印响应信息
#define print_nonnull(name, field)                                \
do {                                                              \
    if (properties-> field) {                                     \
        printf("%s: %s\n", name, properties->field);              \
    }                                                             \
} while (0)
print_nonnull("request_id", request_id);
print_nonnull("request_id2", request_id2);
print_nonnull("content_type", content_type);
if (properties->content_length) {
    printf("content_length: %llu\n", properties->content_length);
}
print_nonnull("server", server);
print_nonnull("ETag", etag);
print_nonnull("expiration", expiration);
print_nonnull("website_redirect_location", website_redirect_location);
print_nonnull("version_id", version_id);
print_nonnull("allow_origin", allow_origin);
print_nonnull("allow_headers", allow_headers);
print_nonnull("max_age", max_age);
print_nonnull("allow_methods", allow_methods);
print_nonnull("expose_headers", expose_headers);
print_nonnull("storage_class", storage_class);
print_nonnull("server_side_encryption", server_side_encryption);
print_nonnull("kms_key_id", kms_key_id);
print_nonnull("customer_algorithm", customer_algorithm);
print_nonnull("customer_key_md5", customer_key_md5);
print_nonnull("bucket_location", bucket_location);
print_nonnull("obs_version", obs_version);
print_nonnull("restore", restore);
print_nonnull("obs_object_type", obs_object_type);
print_nonnull("obs_next_append_position", obs_next_append_position);
print_nonnull("obs_head_epid", obs_head_epid);
print_nonnull("reserved_indicator", reserved_indicator);
int i;
for (i = 0; i < properties->meta_data_count; i++) {
    printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
        properties->meta_data[i].value);
}
return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
if (callback_data) {
    obs_status *ret_status = (obs_status *)callback_data;
    *ret_status = status;
}
else {
    printf("Callback_data is NULL");
}
if (error && error->message) {
    printf("Error Message: \n   %s\n", error->message);
}
if (error && error->resource) {
    printf("Error Resource: \n  %s\n", error->resource);
}
if (error && error->further_details) {
    printf("Error further_details: \n   %s\n", error->further_details);
}
if (error && error->extra_details_count) {
    int i;
    for (i = 0; i < error->extra_details_count; i++) {
```



```
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData)
{
    int urlLen = 0;
    tempAuthResult * ptrResult = (tempAuthResult *)callbackData;
    urlLen = strlen(tempAuthUrl);
    strcpy_s(ptrResult->tmpAuthUrl, 1024, tempAuthUrl);
    strcpy_s(ptrResult->actualHeaders, 1024, tempAuthActualHeaders);
}
```

代码示例四：生成用于列举对象的临时授权 URL

以下示例展示如何生成用于列举对象的临时授权URL：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct __tempAuthResult
{
    char tmpAuthUrl[1024];
    char actualHeaders[1024];
}tempAuthResult;
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData);
int main()
{
    // 以下示例展示如何生成列举对象的URL：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    temp_auth_configure tempauth;
    tempAuthResult  ptrResult;
    memset(&ptrResult, 0, sizeof(tempAuthResult));
    //回调数据
    tempauth.callback_data = (void *)(&ptrResult);
    // 有效时间
    tempauth.expires = 10;
    // 回调函数 返回生成的临时URL
```



```
    tempauth.temp_auth_callback = &tempAuthCallBack_getResult;
    options.temp_auth = &tempauth;
    // 回调函数赋值
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* prefix = "example-prefix";
    char* next_marker = "example-next-marker";
    char* delimiter = "/";
    //列举对象的最大数目
    int maxkeys = 100;
    obs_list_objects_handler list_bucket_objects_handler =
    {
        response_handler,
        NULL
    };
    list_bucket_objects(&options, prefix, next_marker, delimiter, maxkeys,
        &list_bucket_objects_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("the temporary signature url of list objects generated successfully. \n"
            "the temporary signature url is %s. \n"
            "the actualHeaders are %s. \n", ptrResult.tmpAuthUrl, ptrResult.actualHeaders);
    }
    else
    {
        printf(" the temporary signature url of list objects generation failed(%s).\n",
obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                              \
    do {                                                \
        if (properties-> field) {                               \
            printf("%s: %s\n", name, properties->field);        \
        }                                                       \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
```



```
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回
调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char
*tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData)
{
    int urlLen = 0;
    tempAuthResult * ptrResult = (tempAuthResult *)callbackData;
    urlLen = strlen(tempAuthUrl);
    strcpy_s(ptrResult->tmpAuthUrl, 1024, tempAuthUrl);
```



```
    strcpy_s(ptrResult->actualHeaders, 1024, tempAuthActualHeaders);
}
```

代码示例五：生成用于删除对象的临时授权 URL

以下示例展示如何生成用于删除对象的临时授权URL：

```
#include "eSDKOBS.h"
#include <stdio.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data);
typedef struct __tempAuthResult
{
    char tmpAuthUrl[1024];
    char actualHeaders[1024];
}tempAuthResult;
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char *tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData);
int main()
{
    // 以下示例展示如何生成删除对象的URL：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    // 填写Bucket名称，例如example-bucket-name。
    char * bucketName = "example-bucket-name";
    options.bucket_options.bucket_name = bucketName;
    temp_auth_configure tempauth;
    tempAuthResult  ptrResult;
    memset(&ptrResult, 0, sizeof(tempAuthResult));
    //回调数据
    tempauth.callback_data = (void *)(&ptrResult);
    // 有效时间
    tempauth.expires = 10;
    // 回调函数 返回生成的临时URL
    tempauth.temp_auth_callback = &tempAuthCallBack_getResult;
    options.temp_auth = &tempauth;
    // 回调函数赋值
    obs_response_handler response_handler =
    {
        &response_properties_callback,
        &response_complete_callback
    };
    obs_status ret_status = OBS_STATUS_BUTT;
    char* key = "example-object-key";
    char* versionid = NULL;
    // 要删除的对象的信息
    obs_object_info object_info;
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = key;
    object_info.version_id = versionid;
    delete_object(&options, &object_info, &response_handler, &ret_status);
    // 判断请求是否成功
    if (ret_status == OBS_STATUS_OK) {
        printf("the temporary signature url of put object generated successfully. \n"
```



```
            "the temporary signature url is %s. \n"
            "the actualHeaders are %s. \n", ptrResult.tmpAuthUrl, ptrResult.actualHeaders);
    }
    else
    {
        printf(" the temporary signature url of put object generation failed(%s).\n",
obs_get_status_name(ret_status));
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                  \
        if (properties-> field) {                             \
            printf("%s: %s\n", name, properties->field);          \
        }                                                 \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
```



```
    return OBS_STATUS_OK;
}
// 结束回调函数，可以在这个回调中把obs_status和obs_error_details的内容记录到callback_data(用户自定义回调数据)中
void response_complete_callback(obs_status status, const obs_error_details *error, void *callback_data)
{
    if (callback_data) {
        obs_status *ret_status = (obs_status *)callback_data;
        *ret_status = status;
    }
    else {
        printf("Callback_data is NULL");
    }
    if (error && error->message) {
        printf("Error Message: \n   %s\n", error->message);
    }
    if (error && error->resource) {
        printf("Error Resource: \n  %s\n", error->resource);
    }
    if (error && error->further_details) {
        printf("Error further_details: \n   %s\n", error->further_details);
    }
    if (error && error->extra_details_count) {
        int i;
        for (i = 0; i < error->extra_details_count; i++) {
            printf("Error Extra Detail(%d):\n   %s:%s\n", i, error->extra_details[i].name,
                error->extra_details[i].value);
        }
    }
    if (error && error->error_headers_count) {
        int i;
        for (i = 0; i < error->error_headers_count; i++) {
            const char *errorHeader = error->error_headers[i];
            printf("Error Headers(%d):\n    %s\n", i, errorHeader == NULL ? "NULL Header" : errorHeader);
        }
    }
}
void tempAuthCallBack_getResult(char *tempAuthUrl, uint64_t tempAuthUrlLen, char *tempAuthActualHeaders,
    uint64_t tempAuthActualHeadersLen, void *callbackData)
{
    int urlLen = 0;
    tempAuthResult * ptrResult = (tempAuthResult *)callbackData;
    urlLen = strlen(tempAuthUrl);
    strcpy_s(ptrResult->tmpAuthUrl, 1024, tempAuthUrl);
    strcpy_s(ptrResult->actualHeaders, 1024, tempAuthActualHeaders);
}
```

17.3 c sdk 通过自定义域名访问 OBS(C SDK)

须知

开发过程中，您有任何问题可以在github上**提交issue**，或者在**华为云对象存储服务论坛**中发帖求助。

使用自定义域名访问服务端之前，需要先在console界面配置自定义域名。

自定义域名访问介绍与配置

当以自定义域名访问OBS桶时，需要先将该自定义域名同对应OBS桶访问域名进行绑定，相关配置请参见**自定义域名绑定简介**，**自定义域名绑定配置**。

注意

当在自定义域名上配置了CDN加速服务，即自定义域名为CDN服务的加速域名时，需要额外对CDN服务进行配置，以保证可以正常使用自定义域名访问OBS服务。

以华为云CDN服务为例，相关配置如下所示：

步骤1 登录华为云CDN服务，从CDN服务左侧列表中选择域名管理项，在该项中可以查看到所有配置的CDN服务域名信息。

步骤2 配置源站。单击要使用的自定义域名项，进入域名配置界面，编辑源站配置，选择主源站类型为源站域名类型，对应源站为要访问的OBS桶域名。

步骤3 配置回源HOST。回源HOST必须指定为加速域名即访问OBS服务时访问的自定义域名，否则可能会出现回源鉴权失败的问题。

----结束

通过SDK，使用自定义域名访问OBS，可以完全复用对应接口的示例，只需要注意在设置option的时候，按如下方式设置：

```
option.bucket_options.useCname = true;
option.bucket_options.host_name = "yourCustomDomain";
```

代码示例一：通过自定义域名上传对象

以下示例展示如何通过自定义域名上传对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data);
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct put_file_object_callback_data
{
    FILE *infile;
    uint64_t content_length;
    obs_status ret_status;
} put_file_object_callback_data;
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data);
int main()
{
    // 以下示例展示如何在使用自定义域名时通过put_object函数上传本地文件：：
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "";
    options.bucket_options.bucket_name = bucketName;
    // 设置自定义域名
    options.bucket_options.useCname = true;
    options.bucket_options.host_name = "example.obs.test.cname.com";
    // 初始化上传对象属性
    obs_put_properties put_properties;
    init_put_properties(&put_properties);
    // 上传对象名
    char *key = "example_get_file_test";
    // 上传的文件
    char file_name[256] = "./example_local_file_test.txt";
    uint64_t content_length = 0;
    // 初始化存储上传数据的结构体
    put_file_object_callback_data data;
    memset(&data, 0, sizeof(put_file_object_callback_data));
    // 打开文件，并获取文件长度
    content_length = open_file_and_get_length(file_name, &data);
    // 设置回调函数
    obs_put_object_handler putobjectHandler =
    {
        { &response_properties_callback, &put_file_complete_callback },
        &put_file_data_callback
    };
    put_object(&options, key, content_length, &put_properties, 0, &putobjectHandler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("put object from file successfully. \n");
    }
    else
    {
        printf("put object failed(%s).\n",
            obs_get_status_name(data.ret_status));
    }
    if (data.infile != NULL) {
        fclose(data.infile);
```



```
    }
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
int put_file_data_callback(int buffer_size, char *buffer,
    void *callback_data)
{
    put_file_object_callback_data *data =
        (put_file_object_callback_data *)callback_data;
```



```
    int ret = 0;
    if (data->content_length) {
        int toRead = ((data->content_length > (unsigned)buffer_size) ?
            (unsigned)buffer_size : data->content_length);
        ret = fread(buffer, 1, toRead, data->infile);
    }
    uint64_t originalContentLength = data->content_length;
    data->content_length -= ret;
    if (data->content_length) {
        printf("%llu bytes remaining ", (unsigned long long)data->content_length);
        printf("(%d%% complete) ...\n",
            (int)(((originalContentLength - data->content_length) * 100) / originalContentLength));
    }
    return ret;
}
void put_file_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    put_file_object_callback_data *data = (put_file_object_callback_data *)callback_data;
    data->ret_status = status;
}
uint64_t open_file_and_get_length(char *localfile, put_file_object_callback_data *data)
{
    uint64_t content_length = 0;
    const char *body = 0;
    if (!content_length)
    {
        struct stat statbuf;
        if (stat(localfile, &statbuf) == -1)
        {
            fprintf(stderr, "\nERROR: Failed to stat file %s: ",
                localfile);
            return 0;
        }
        content_length = statbuf.st_size;
    }
    if (!(data->infile = fopen(localfile, "rb")))
    {
        fprintf(stderr, "\nERROR: Failed to open input file %s: ",
            localfile);
        return 0;
    }
    data->content_length = content_length;
    return content_length;
}
```

代码示例二：通过自定义域名下载对象

以下示例展示如何通过自定义域名下载对象：

```
#include "eSDKOBS.h"
#include <stdio.h>
#include <sys/stat.h>
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data);
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data);
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data);
typedef struct get_object_callback_data
{
    FILE *outfile;
    obs_status ret_status;
}get_object_callback_data;
FILE * write_to_file(char *localfile);
int main()
{
    // 以下示例展示如何在使用自定义域名时通过get_object函数下载对象到本地文件：
```

```
    // 在程序入口调用obs_initialize方法来初始化网络、内存等全局资源。
    obs_initialize(OBS_INIT_ALL);
    obs_options options;
    // 创建并初始化options，该参数包括访问域名(host_name)、访问密钥（access_key_id和
acces_key_secret）、桶名(bucket_name)、桶存储类别(storage_class)等配置信息
    init_obs_options(&options);
    // host_name填写桶所在的endpoint, 此处以华北-北京四为例，其他地区请按实际情况填写。
    options.bucket_options.host_name = "obs.cn-north-4.myhuaweicloud.com";
    // 认证用的ak和sk硬编码到代码中或者明文存储都有很大的安全风险，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；
    // 本示例以ak和sk保存在环境变量中为例，运行本示例前请先在本地环境中设置环境变量ACCESS_KEY_ID和
SECRET_ACCESS_KEY。
    options.bucket_options.access_key = getenv("ACCESS_KEY_ID");
    options.bucket_options.secret_access_key = getenv("SECRET_ACCESS_KEY");
    char * bucketName = "";
    options.bucket_options.bucket_name = bucketName;
    // 设置自定义域名
    options.bucket_options.useCname = true;
    options.bucket_options.host_name = "example.obs.test.cname.com";
    // 设置对象下载到本地的文件名
    char *file_name = "./example_get_file_test";
    obs_object_info object_info;
    // 设置下载的对象
    memset(&object_info, 0, sizeof(obs_object_info));
    object_info.key = "example_get_file_test";
    object_info.version_id = NULL;
    //根据业务需要设置存放下载对象数据的结构
    get_object_callback_data data;
    data.ret_status = OBS_STATUS_BUTT;
    data.outfile = write_to_file(file_name);
    // 定义范围下载参数
    obs_get_conditions getcondition;
    memset(&getcondition, 0, sizeof(obs_get_conditions));
    init_get_properties(&getcondition);
    // 下载起始位置，默认0，从对象0字节位置开始下载
    getcondition.start_byte = 0;
    // 下载长度，默认0，一直下载到对象尾
    // getcondition.byte_count = 0;
    // 定义下载的回调函数
    obs_get_object_handler get_object_handler =
    {
        { &response_properties_callback,
          &get_object_complete_callback},
        &get_object_data_callback
    };
    get_object(&options, &object_info, &getcondition, 0, &get_object_handler, &data);
    if (OBS_STATUS_OK == data.ret_status) {
        printf("get object successfully. \n");
    }
    else
    {
        printf("get object faied(%s).\n", obs_get_status_name(data.ret_status));
    }
    fclose(data.outfile);
    // 释放分配的全局资源
    obs_deinitialize();
}
// 响应回调函数，可以在这个回调中把properties的内容记录到callback_data(用户自定义回调数据)中
obs_status response_properties_callback(const obs_response_properties *properties, void *callback_data)
{
    if (properties == NULL)
    {
        printf("error! obs_response_properties is null!");
        if (callback_data != NULL)
        {
            obs_sever_callback_data *data = (obs_sever_callback_data *)callback_data;
            printf("server_callback buf is %s, len is %llu",
                data->buffer, data->buffer_len);
            return OBS_STATUS_OK;
```



```
        }
        else {
            printf("error! obs_sever_callback_data is null!");
            return OBS_STATUS_OK;
        }
    }
    // 打印响应信息
#define print_nonnull(name, field)                                \
    do {                                                          \
        if (properties-> field) {                                 \
            printf("%s: %s\n", name, properties->field);          \
        }                                                         \
    } while (0)
    print_nonnull("request_id", request_id);
    print_nonnull("request_id2", request_id2);
    print_nonnull("content_type", content_type);
    if (properties->content_length) {
        printf("content_length: %llu\n", properties->content_length);
    }
    print_nonnull("server", server);
    print_nonnull("ETag", etag);
    print_nonnull("expiration", expiration);
    print_nonnull("website_redirect_location", website_redirect_location);
    print_nonnull("version_id", version_id);
    print_nonnull("allow_origin", allow_origin);
    print_nonnull("allow_headers", allow_headers);
    print_nonnull("max_age", max_age);
    print_nonnull("allow_methods", allow_methods);
    print_nonnull("expose_headers", expose_headers);
    print_nonnull("storage_class", storage_class);
    print_nonnull("server_side_encryption", server_side_encryption);
    print_nonnull("kms_key_id", kms_key_id);
    print_nonnull("customer_algorithm", customer_algorithm);
    print_nonnull("customer_key_md5", customer_key_md5);
    print_nonnull("bucket_location", bucket_location);
    print_nonnull("obs_version", obs_version);
    print_nonnull("restore", restore);
    print_nonnull("obs_object_type", obs_object_type);
    print_nonnull("obs_next_append_position", obs_next_append_position);
    print_nonnull("obs_head_epid", obs_head_epid);
    print_nonnull("reserved_indicator", reserved_indicator);
    int i;
    for (i = 0; i < properties->meta_data_count; i++) {
        printf("x-obs-meta-%s: %s\n", properties->meta_data[i].name,
            properties->meta_data[i].value);
    }
    return OBS_STATUS_OK;
}
obs_status get_object_data_callback(int buffer_size, const char *buffer,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    size_t wrote = fwrite(buffer, 1, buffer_size, data->outfile);
    return ((wrote < (size_t)buffer_size) ?
        OBS_STATUS_AbortedByCallback : OBS_STATUS_OK);
}
void get_object_complete_callback(obs_status status,
    const obs_error_details *error,
    void *callback_data)
{
    get_object_callback_data *data = (get_object_callback_data *)callback_data;
    data->ret_status = status;
}
FILE * write_to_file(char *localfile)
{
    FILE *outfile = 0;
    if (localfile) {
        struct stat buf;
        if (stat(localfile, &buf) == -1) {
```



```
            outfile = fopen(localfile, "wb");
        }
        else {
            outfile = fopen(localfile, "a");
        }
        if (!outfile) {
            fprintf(stderr, "\nERROR: Failed to open output file %s: ",
                localfile);
            return outfile;
        }
    } else {
        fprintf(stderr, "\nERROR: Failed to open output file, it's NULL, write object to stdout.",
            localfile);
        outfile = stdout;
    }
    return outfile;
}
```


18 异常处理(C SDK)

18.1 OBS 服务端错误码(C SDK)

在向OBS服务端发出请求后，如果遇到错误，会在响应中包含响应的错误码描述错误信息。详细的错误码及其对应的描述和HTTP状态码见下表：

HTTP状态码	错误码	错误信息	处理措施
301 Moved Permanently	PermanentRedirect	尝试访问的桶必须使用指定的地址，请将以后的请求发送到这个地址。	按照返回的重定向地址发送请求。
301 Moved Permanently	WebsiteRedirect	Website请求缺少bucketName。	携带桶名后重试。
307 Moved Temporarily	TemporaryRedirect	临时重定向，当DNS更新时，请求将被重定向到桶。	会自动重定向，也可以将请求发送到重定向地址。
400 Bad Request	BadDigest	客户端指定的对象内容的MD5值与系统接收到的内容MD5值不一致。	检查头域中携带的MD5与消息体计算出来的MD5是否一致。
400 Bad Request	BadDomainName	域名不合法。	使用合法的域名。
400 Bad Request	BadRequest	请求参数不合法。	根据返回的错误消息体提示进行修改。
400 Bad Request	CustomDomainAreadyExist	配置了已存在的域。	已经配置过了，不需要再配置。
400 Bad Request	CustomDomainNotExist	删除不存在的域。	未配置或已经删除，无需删除。

HTTP状态码	错误码	错误信息	处理措施
400 Bad Request	EntityTooLarge	用户POST上传的对象大小超过了条件允许的最大大小。	修改POST上传的policy中的条件或者减少对象大小。
400 Bad Request	EntityTooSmall	用户POST上传的对象大小小于条件允许的最小大小。	修改POST上传的policy中的条件或者增加对象大小。
400 Bad Request	IllegalLocationConstraintException	用户未带Location在非默认Region创桶。	请求发往默认Region创桶或带非默认Region的Location创桶。
400 Bad Request	IncompleteBody	由于网络原因或其他问题导致请求体未接受完整。	重新上传对象。
400 Bad Request	IncorrectNumberOfFilesInPostRequest	每个POST请求都需要带一个上传的文件。	带上一个上传文件。
400 Bad Request	InvalidArgument	无效的参数。	根据返回的错误消息体提示进行修改。
400 Bad Request	InvalidBucket	请求访问的桶已不存在。	更换桶名。
400 Bad Request	InvalidBucketName	请求中指定的桶名无效，超长或带不允许的特殊字符。	更换桶名。
400 Bad Request	InvalidEncryptionAlgorithmError	错误的加密算法。下载SSE-C加密的对象，携带的加密头域错误，导致不能解密。	携带正确的加密头域下载对象。
400 Bad Request	InvalidLocationConstraint	创建桶时，指定的Location不合法或不存在。	指定正确的Location创桶。
400 Bad Request	InvalidPart	一个或多个指定的段无法找到。这些段可能没有上传，或者指定的entity tag与段的entity tag不一致。	按照正确的段和entity tag合并段。
400 Bad Request	InvalidPartOrder	段列表的顺序不是升序，段列表必须按段号升序排列。	按段号升序排列后重新合并。

HTTP状态码	错误码	错误信息	处理措施
400 Bad Request	InvalidPolicyDocument	表单中的内容与策略文档中指定的条件不一致。	根据返回的错误消息体提示修改构造表单的policy重试。
400 Bad Request	InvalidRedirectLocation	无效的重定向地址。	指定正确的地址。
400 Bad Request	InvalidRequest	无效请求。	根据返回的错误消息体提示进行修改。
400 Bad Request	InvalidRequestBody	请求体无效，需要消息体的请求没有上传消息体。	按照正确的格式上传消息体。
400 Bad Request	InvalidTargetBucketForLogging	delivery group对目标桶无ACL权限。	对目标桶配置ACL权限后重试。
400 Bad Request	KeyTooLongError	提供的Key过长。	使用较短的Key。
400 Bad Request	KMS.DisabledException	SSE-KMS加密方式下，主密钥被禁用。	更换密钥后重试，或联系技术支持。
400 Bad Request	KMS.NotFoundException	SSE-KMS加密方式下，主密钥不存在。	携带正确的主密钥重试。
400 Bad Request	MalformedACLError	提供的XML格式错误，或者不符合要求的格式。	使用正确的XML格式重试。
400 Bad Request	MalformedError	请求中携带的XML格式不正确。	使用正确的XML格式重试。
400 Bad Request	MalformedLoggingStatus	Logging的XML格式不正确。	使用正确的XML格式重试。
400 Bad Request	MalformedPolicy	Bucket policy检查不通过。	根据返回的错误消息体提示结合桶policy的要求进行修改。
400 Bad Request	MalformedQuotaError	Quota的XML格式不正确。	使用正确的XML格式重试。
400 Bad Request	MalformedXML	当用户发送了一个配置项的错误格式的XML会出现这样的错误。	使用正确的XML格式重试。
400 Bad Request	MaxMessageLengthExceeded	拷贝对象，带请求消息体。	拷贝对象不带消息体重试。

HTTP状态码	错误码	错误信息	处理措施
400 Bad Request	MetadataTooLarge	元数据消息头超过了允许的最大元数据大小。	减少元数据消息头。
400 Bad Request	MissingRegion	请求中缺少Region信息，且系统无默认Region。	请求中携带Region信息。
400 Bad Request	MissingRequestBodyError	当用户发送一个空的XML文档作为请求时会发生。	提供正确的XML文档。
400 Bad Request	MissingRequiredHeader	请求中缺少必要的头域。	提供必要的头域。
400 Bad Request	MissingSecurityHeader	请求缺少一个必须的头。	提供必要的头域。
400 Bad Request	TooManyBuckets	用户拥有的桶的数量达到了系统的上限，并且请求试图创建一个新桶。	删除部分桶后重试。
400 Bad Request	TooManyCustomDomains	配置了过多的用户域	删除部分用户域后重试。
400 Bad Request	TooManyWrongSignature	因高频错误请求被拒绝服务。	更换正确的Access Key后重试。
400 Bad Request	UnexpectedContent	该请求需要消息体而客户端没带，或该请求不需要消息体而客户端带了。	根据说明重试。
400 Bad Request	UserKeyMustBeSpecified	该操作只有特殊用户可使用。	请联系技术支持。
403 Forbidden	AccessDenied	拒绝访问，请求没有携带日期头域或者头域格式错误。	请求携带正确的日期头域。
403 Forbidden	AccessForbidden	权限不足，桶未配置CORS或者CORS规则不匹配。	修改桶的CORS配置，或者根据桶的CORS配置发送匹配的OPTIONS请求。
403 Forbidden	AllAccessDisabled	用户无权限执行某操作。桶名为禁用关键字。	更换桶名。
403 Forbidden	DeregisterUserId	用户已经注销。	充值或重新开户。

HTTP状态码	错误码	错误信息	处理措施
403 Forbidden	InArrearOrInsufficientBalance	用户欠费或余额不足而没有权限进行某种操作。	充值。
403 Forbidden	InsufficientStorageSpace	存储空间不足。	超过配额限制，增加配额或删除部分对象。
403 Forbidden	InvalidAccessKeyId	系统记录中不存在客户提供的Access Key Id。	携带正确的Access Key Id。
403 Forbidden	NotSignedUp	你的账户还没有在系统中注册，必须先在系统中注册了才能使用该账户。	先注册OBS服务。
403 Forbidden	RequestTimeTooSkewed	请求的时间与服务器的时间相差太大。	检查客户端时间是否与当前时间相差太大。
403 Forbidden	SignatureDoesNotMatch	请求中带的签名与系统计算得到的签名不一致。	检查你的Secret Access Key和签名计算方法。
403 Forbidden	Unauthorized	用户未实名认证。	请实名认证后重试。
404 Not Found	NoSuchBucket	指定的桶不存在。	先创桶再操作。
404 Not Found	NoSuchBucketPolicy	桶policy不存在。	先配置桶policy。
404 Not Found	NoSuchCORSConfiguration	CORS配置不存在。	先配置CORS。
404 Not Found	NoSuchCustomDomain	请求的用户域不存在。	先设置用户域。
404 Not Found	NoSuchKey	指定的Key不存在。	先上传对象。
404 Not Found	NoSuchLifecycleConfiguration	请求的LifeCycle不存在。	先配置LifeCycle。
404 Not Found	NoSuchUpload	指定的多段上传不存在。Upload ID不存在，或者多段上传已经终止或完成。	使用存在的段或重新初始化段。
404 Not Found	NoSuchVersion	请求中指定的version ID与现存的所有版本都不匹配。	使用正确的version ID。

HTTP状态码	错误码	错误信息	处理措施
404 Not Found	NoSuchWebsiteConfiguration	请求的Website不存在。	先配置Website。
405 Method Not Allowed	MethodNotAllowed	指定的方法不允许操作在请求的资源上。 对应返回的Message为：Specified method is not supported.	方法不允许。
408 Request Timeout	RequestTimeout	用户与Server之间的socket连接在超时时间内没有进行读写操作。	检查网络后重试，或联系技术支持。
409 Conflict	BucketAlreadyExists	请求的桶名已经存在。桶的命名空间是系统中所有用户共用的，选择一个不同的桶名再重试一次。	更换桶名。
409 Conflict	BucketAlreadyOwnedByYou	发起该请求的用户已经创建过了这个名字的桶，并拥有这个桶。	不需要再创桶了。
409 Conflict	BucketNotEmpty	用户尝试删除的桶不为空。	先删除桶中对象，然后再删桶。
409 Conflict	InvalidBucketState	无效的桶状态，配置跨Region复制后不允许关闭桶多版本。	不关闭桶的多版本或取消跨Region复制。
409 Conflict	OperationAborted	另外一个冲突的操作当前正作用在这个资源上，请重试。	等待一段时间后重试。
409 Conflict	ServiceNotSupported	请求的方法服务端不支持。	服务端不支持，请联系技术支持。
411 Length Required	MissingContentLength	必须要提供HTTP消息头中的Content-Length字段。	提供Content-Length消息头。
412 Precondition Failed	PreconditionFailed	用户指定的先决条件中至少有一项没有包含。	根据返回消息体中的Condition提示进行修改。

HTTP状态码	错误码	错误信息	处理措施
416 Client Requested Range Not Satisfiable	InvalidRange	请求的range不可获得。	携带正确的range重试。
500 Internal Server Error	InternalError	系统遇到内部错误，请重试。	请联系技术支持。
501 Not Implemented	ServiceNotImplemented	请求的方法服务端没有实现。	当前不支持，请联系技术支持。
503 Service Unavailable	ServiceUnavailable	服务器过载或者内部错误异常。	等待一段时间后重试，或联系技术支持。
503 Service Unavailable	SlowDown	请降低请求频率。	请降低请求频率。

18.2 SDK 错误处理(C SDK)

SDK错误返回包含： SDK检查函数参数返回的错误和OBS服务端返回的错误。

SDK错误处理信息：

- obs_status：错误码。
- obs_get_status_name()：获取错误描述。
- obs_status_is_retryable()：确认错误码是否需要业务重试。

18.3 日志分析(C SDK)

日志路径

OBS C SDK的日志路径是通过OBS.ini中LogPath字段指定的，默认存放于与C SDK动态库lib目录同级的logs目录中。定位问题只需要查看同级logs目录下运行日志eSDK-OBS-API-*-C.run.log或者obs-sdk-c.run.log。

OBS.ini文件应与动态库（libeSDKLogAPI.so）同一目录。

日志内容格式

SDK日志格式为：日志时间|日志级别|线程号|日志内容。示例如下：

```
运行日志：
2018-05-15 22:22:54 803| INFO|[140677572568864]|request_perform start
```

日志级别

当系统出现问题需要定位且当前的日志无法满足要求时，可以通过修改日志的级别来获取更多的信息。其中DEBUG(0)日志信息最丰富，ERROR(3)日志信息最少。

具体说明如下：

- DEBUG(0)：调试级别，如果设置为这个级别，除了打印INFO级别的信息外，还将打印其它帮助调试的信息等。
- INFO(1)：信息级别，如果设置为这个级别，除了打印WARN级别的信息外，还将打印OBS接口的调用过程和关键信息等。
- WARN(2)：告警级别，如果设置为这个级别，除了打印ERROR级别的信息外，还将打印一些关键事件的信息，如curl_global_init初始化失败等。
- ERROR(3)：错误级别，如果设置为这个级别，仅打印发生异常时的错误信息。

设置方式

修改lib目录下的OBS.ini，配置日志的大小、个数以及级别（其中*_Run参数的配置为最常用的配置项）：

```
;Every line must be less than 1024
[LogConfig]
;Log Size: unit=KB, 10MB = 10KB * 1024 = 10240KB
LogSize_Interface=10240
LogSize_Operation=10240
LogSize_Run=10240
;Log Num
LogNum_Interface=10
LogNum_Operation=10
LogNum_Run=10
;Log level: debug = 0,info = 1,warn = 2,error = 3
LogLevel_Interface=0
LogLevel_Operation=0
LogLevel_Run=0
;LogFilePermission
LogFilePermission=0600
[ProductConfig]
;Product Name
sdkname=eSDK-OBS-API-Linux-C
[LogPath]
;Log Path is relative to the path of configuration file
LogPath=../logs
```

其他日志相关配置

Windows下，OBS.ini中LogPath字段可以以wchar_t格式读取，需要通过设置文件路径编码来实现，案例如下：

set_file_path_code(UNICODE_CODE);//默认是ANSI_CODE

设置后，以下方法的本地文件路径也需要是wchar_t类型（参数是以char*格式传入，内部处理逻辑是wchar_t）

表 18-1

影响函数	说明
download_file	参数download_file_config的downLoad_file、check_point_file成员需要是wchar_t类型，以char*格式传入
upload_file	参数upload_file_config的upload_file、check_point_file成员需要是wchar_t类型，以char*格式传入
set_obs_log_path	参数log_path需要是wchar_t类型，以char*格式传入

19 常见问题(C SDK)

19.1 代理设置失效(C SDK)

- **sdk Windows端 demo中设置代理时出现如下问题，程序报错且代理设置失败。**

  ```
  Run-Time Check Failure #2 - Stack around the variable 'ret_status' was corrupted.
  ```

  问题根因：某些sdk版本demo头文件eSDKOBS.h与sdk的eSDKOBS.h未同步更新，导致option中设置的代理失效。

  解决方法：

  a. 将“yourSDKpath\source\eSDK_OBS_API\eSDK_OBS_API_C++\inc\eSDKOBS.h”替换为“yourSDKpath\source\eSDK_OBS_API\eSDK_OBS_API_C++\build\obs\demo\eSDKOBS.h”。

  b. demo做如下改动来适配eSDKOBS.h的更改（适配过程以3.22.7版本为例，其他版本可能略有不同）。

  在文件yourSDKpath\source\eSDK_OBS_API\eSDK_OBS_API_C++\build\obs\demo\ demo_windows.cpp中4749行新增obs_upload_file_server_callback server_callback；同时4750行中，函数upload_file第四个参数后增加, server_callback，如下图：

  ```
  };
  obs_upload_file_server_callback server_callback;
  upload_file(&option, key, 0, &uploadFileInfo, server_callback, &Handler, 0);
  if (statusG == OBS_STATUS_OK) {
      printf("test upload file successfully. \n");
  }
  else
  {
  printf("test upload file faied.\n");
  printError();
  return ;
  }
  }
  ```

- **设置了代理还是连接失败。**

  问题根因：可能是因为sdk的request.c的get_api_version函数中未设置代理。

  解决办法：

可以参考sdk的request.c的 setup_curl函数中设置代理的方式在get_api_version函数中添加向curl中设置代理（CURLOPT_PROXY项与CURLOPT_PROXYUSERPWD项）的逻辑进行修复。

19.2 代码示例常见问题(C SDK)

v3.23.9版本反映的常见代码示例问题

问题1：找不到put_file_object_callback_data 代码

解决办法：

3.23.9版本中代码示例部分变量或函数缺失，可以在**eSDK_OBS_API_C++_Demo**中查找对应的定义，然后复制到业务代码中调用。

19.3 在 Linux 环境中访问 OBS 报错：requires a valid Date or x-obs-date header(C SDK)

问题根因：

Linux环境变量（env命令）中，LANG变量值为zh_CN.UTF-8，请求头Date会包含中文，例如“三，04 9月 2024 10:41:39 GMT”。

解决方案：

将环境变量LANG设置为en_US.UTF-8，代码示例如下：

```
#include <locale.h>
setlocale(LC_TIME, "en_GB.UTF-8");
```

19.4 如何获取账号 ID 和用户 ID?

获取账号、IAM 用户、项目的名称和 ID

- **从控制台获取账号名、账号ID、用户名、用户ID、项目名称、项目ID**
  a. 在华为云首页右上角，单击“控制台”。
  b. 在右上角的用户名中选择“我的凭证”。

图 19-1 进入我的凭证

c. 在“我的凭证”界面，API凭证页签中，查看账号名、账号ID、用户名、用户ID、项目名称、项目ID。

每个区域的项目ID有所不同，需要根据业务所在的区域获取对应的项目ID。

图 19-2 查看账号名、账号 ID、用户名、用户 ID、项目名称、项目 ID

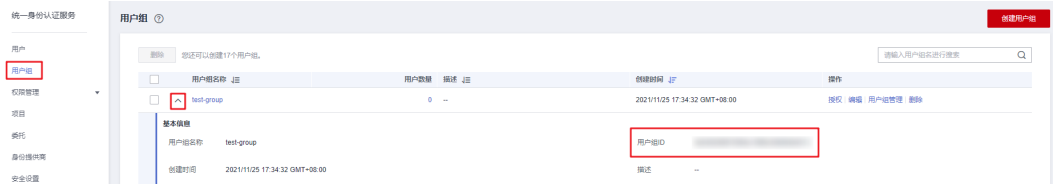
- **调用API获取用户ID、项目ID**
  - 获取用户ID请参考：**管理员查询IAM用户列表**。
  - 获取项目ID请参考：**查询指定条件下的项目列表**。

获取用户组名称和 ID

步骤1 登录华为云，进入IAM控制台，选择“用户组”页签。

步骤2 单击需要查询的用户组前的下拉框，即可查询用户组名称、用户组ID。

图 19-3 查询用户组名称、用户组 ID



----结束

获取区域 ID

步骤1 登录华为云，进入IAM控制台，选择“项目”页签。

步骤2 “项目”列的内容即为所属区域对应的ID。

图 19-4 查看区域 ID

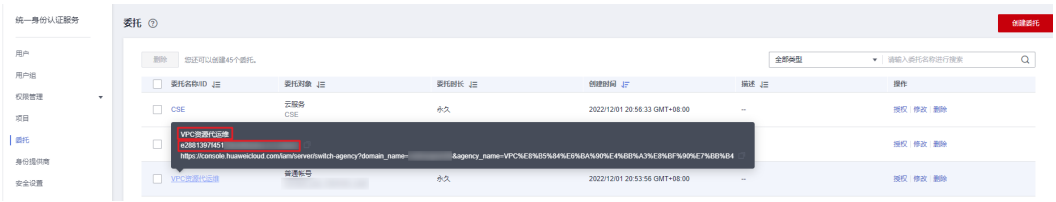
----结束

获取委托名称和 ID

步骤1 登录华为云，进入IAM控制台，选择“委托”页签。

步骤2 鼠标移动到需要查询名称和ID的委托上，黑色框中出现的第一行为委托名称，第二行为委托ID。

图 19-5 查看委托 ID



----结束