对象存储服务

Android SDK 开发指南

文档版本	01
发布日期	2024-12-03

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 SDK 安装

下载地址

- OBS Android SDK最新版本：**OBS Android SDK**

SDK 源码和 API 文档

- SDK源码请参见：**GitHub**
- 接口参考文档地址：**SDK API文档**

兼容性

- 版本修订记录信息：**ChangeLog**。
- 推荐使用的Android系统版本：Android 7.0，8.0，8.1，9.0，10.0版本（HTTP/HTTPS协议）。
- jdk1.8环境建议使用3.21.12版本的OBS Android SDK。
- 命名空间：与旧版本（2.1.x）保持兼容，对外的接口都包含在com.obs.services，com.obs.services.model和com.obs.services.exception三个包中。
- 接口函数：与旧版本（2.1.x）保持兼容。

  说明

  Android 4.4及以下版本只能使用配置HTTP协议访问OBS服务。

混淆配置

```
#okhttp
-dontwarn okhttp3.**
-keep class okhttp3.**{*;}

#okio
-dontwarn okio.**
-keep class okio.**{*;}

#sdk
-keep class com.obs.services.** {*;}
-keep class com.obs.log.** {*;}

#java-xmlbuilder
-keep class com.jamesmurty.utils.**{*;}
```

2 示例程序

OBS Android SDK提供了丰富的示例程序，方便用户参考或直接使用。您可以从OBS Android SDK开发包中获取示例程序，如eSDK_Storage_OBS_<VersionId>_Android.zip，解压后eSDK_Storage_OBS_<VersionId>_Android/samples_android为示例程序。您也可以从下面表格中直接下载示例程序。

示例包括以下内容：

示例代码	说明
BucketOperationsSample	展示了桶相关接口的用法
ObjectOperationsSample	展示了对象相关接口的用法
DownloadSample	展示了下载对象的用法
CreateFolderSample	展示了创建文件夹的用法
DeleteObjectsSample	展示了批量删除对象的用法
ListObjectsSample	展示了列举对象的用法
ListVersionsSample	展示了列举多版本对象的用法
ListObjectsInFolderSample	展示了列举文件夹内对象的用法
ObjectMetaSample	展示了自定义对象元数据的用法
SimpleMultipartUploadSample	展示了分段上传的基本用法
ConcurrentCopyPartSample	展示了分段并发复制大对象的用法
ConcurrentDownloadObjectSample	展示了分段并发下载大对象的用法
ConcurrentUploadPartSample	展示了分段并发上传大对象的用法
RestoreObjectSample	展示了下载归档存储对象的用法
PostObjectSample	展示了表单上传对象的用法
TemporarySignatureSample	展示了使用URL进行授权访问的用法

3 快速入门

3.1 使用前需知

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

- 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。
- 您可以先参考**OBS客户端通用示例**，了解OBS Android SDK接口调用的通用方式。
- 使用OBS客户端进行接口调用操作完成后，没有异常抛出，则表明返回值有效；如果抛出异常，则说明操作失败，此时应可**SDK自定义异常**实例中获取错误信息。
- 使用OBS客户端进行接口调用成功后，均会返回包含响应头信息的**SDK公共响应头**实例（或其子类实例）。
- 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。

3.2 OBS 服务环境搭建

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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/intl/zh-cn。
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“费用与成本”，单击左侧资金管理。
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。

创建访问密钥的操作步骤如下：

1. 登录OBS控制台。
2. 单击页面右上角的用户名，并选择“我的凭证”。
3. 在“我的凭证”页面，单击左侧导航栏的“访问密钥”。
4. 在“访问密钥”页面，单击“新增访问密钥”。
5. 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。
6. 单击“确定”。
7. 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。
8. 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。

----结束

3.3 开发环境准备

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

- 从**Android官网**下载并安装Android Studio最新版本。
- 打开安装好的Android Studio，选择Settings -> Appearance & Behavior -> System Settings -> Android SDK安装应用程序需要支持的SDK Platforms，如Android 5.0 (Lollipop)。

3.4 安装 SDK

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

使用Android Studio自带的gradle下载安装OBS Android SDK，步骤如下：

步骤1 打开Android Studio，单击“Start a new Android Studio project”进入创建工程引导界面。

步骤2 依次填入各配置项（如“Application name”，“Project location”等）直至工程创建成功。

步骤3 在Android Studio中找到新建工程根目录的build.gradle文件（注意不是app文件夹的build.gradle），并在该文件中加入如下配置信息：

```
repositories{
    maven {
        url 'https://mirrors.huaweicloud.com/repository/maven/'
    }
    mavenCentral()
}
```

步骤4 在新建工程的app文件夹build.gradle文件里面找到dependencies节点，在该节点内部新增一行。如下：

```
implementation 'com.huaweicloud:esdk-obs-android:3.24.3'
```

步骤5 打开AndroidManifest.xml文件，加入以下代码片段以配置SDK所需权限：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.GET_TOP_ACTIVITY_INFO" />
```

步骤6 编译工程，通过gradle自动下载SDK，完成集成。

----结束

3.5 获取服务地址

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

- 您可以从**这里**查看OBS当前开通的服务地址和区域信息。

须知

SDK支持带协议名和不带协议名两种方式传入服务地址，例如获取到的服务地址为“your-endpoint”，则初始化OBS客户端时传入的服务地址可以为“http://your-endpoint”、“https://your-endpoint”和“your-endpoint”三种形式。

3.6 初始化 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

向OBS发送任一HTTP/HTTPS请求之前，必须先创建一个ObsClient实例：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 使用访问OBS

// 关闭obsClient
obsClient.close();
```

说明

更多关于OBS客户端初始化的操作请参见“初始化”章节。

日志配置详见**配置SDK日志**

3.7 创建桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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：

```
obsClient.createBucket("bucketname");
```

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、“-”、“.”。
  - 禁止使用类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。
- 更多创建桶的信息，请参见**创建桶**。

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。
- 您可以使用**带参数创建**方式，在创建桶时，指定桶的区域位置。

3.8 上传对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于上传字符串“Hello OBS”到桶名为“bucketname”里，名称为“objectname”。

代码示例如下所示：

```
obsClient.putObject("bucketname", "objectname", new ByteArrayInputStream("Hello OBS".getBytes()));
```

说明

- 更多上传对象的信息，请参见**上传对象**。

3.9 下载对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于下载桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
ObsObject obsObject = obsClient.getObject("bucketname", "objectname");
InputStream content = obsObject.getObjectContent();
if (content != null)
{
    BufferedReader reader = new BufferedReader(new InputStreamReader(content));
    while (true)
    {
        String line = reader.readLine();
        if (line == null)
            break;
        Log.i("GetObject", "\n" + line);
    }
    reader.close();
}
```

说明

- 调用ObsClient.getObject返回一个ObsObject实例，该实例包含对象内容及其属性。
- 调用ObsObject.getObjectContent获取对象输入流，可读取此输入流获取其内容，用完之后请关闭这个流。
- 更多下载对象的信息，请参见**下载对象**。

3.10 列举对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当完成一系列上传对象操作后，可能需要查看桶中包含哪些对象。以下代码展示如何列举指定桶中的对象：

```
ObjectListing objectListing = obsClient.listObjects("bucketname");
for(ObsObject obsObject : objectListing.getObjects()){
    Log.i("ListObjects", " - " + obsObject.getObjectKey() + "  " +  "(size = " +
obsObject.getMetadata().getContentLength() + ")");
}
```

说明

- 调用ObsClient.listObjects返回ObjectListing实例，该实例包含此次listObject请求的返回结果，可通过ObjetListing.getObjects获取所有对象（Object）的描述信息。
- 上面的代码默认列举1000个对象（Object）。
- 更丰富的列举功能，请参见**列举对象**。

3.11 删除对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

本示例用于删除桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
obsClient.deleteObject("bucketname", "objectname");
```

说明

- 本示例仅用于删除单个对象，OBS批量删除对象，需自行遍历构建待批量删除对象的列表。
- 更丰富的删除功能，请参见**删除对象**。

3.12 OBS 客户端通用示例

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

使用OBS客户端进行接口调用操作完成后，没有异常抛出，则表明返回值有效，返回**SDK公共响应头**实例或其子类实例；如果抛出异常，则说明操作失败，此时应从**SDK自定义异常**实例中获取错误信息。

OBS客户端的通用方式用于上传本地“localfile”到桶名为“bucketname”里，名称为“objectname”。

代码示例如下所示：

```
// 您的工程中可以只保留一个全局的ObsClient实例
// ObsClient是线程安全的，可在并发场景下使用
ObsClient obsClient = null;
try
{
    // 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AccessKeyID和SecretAccessKey。
    // 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/intl/zh-cn/usermanual-  ca/ca_01_0003.html
    String ak = System.getenv("ACCESS_KEY_ID");
    String sk = System.getenv("SECRET_ACCESS_KEY_ID");
```



```
    String endPoint = "https://your-endpoint";

    // 创建ObsClient实例
    obsClient = new ObsClient(ak, sk, endPoint);
    // 调用接口进行操作，例如上传对象，其中localfile为待上传的本地文件路径，需要指定到具体的文件名
    HeaderResponse response = obsClient.putObject("bucketname", "objectname", new File("localfile"));
    Log.i("PutObject", response);
}
catch (ObsException e)
{
    Log.e("PutObject", "Response Code: " + e.getResponseCode());
    Log.e("PutObject", "Error Message: " + e.getErrorMessage());
    Log.e("PutObject", "Error Code:       " + e.getErrorCode());
    Log.e("PutObject", "Request ID:      " + e.getErrorRequestId());
    Log.e("PutObject", "Host ID:           " + e.getErrorHostId());
}finally{
    // 关闭ObsClient实例，如果是全局ObsClient实例，可以不在每个方法调用完成后关闭
    // ObsClient在调用ObsClient.close方法关闭后不能再次使用
    if(obsClient != null){
        try
        {
            obsClient.close();
        }
        catch (IOException e)
        {
        }
    }
}
```


4 初始化

4.1 配置密钥

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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**。

获取AK和SK之后，您便可以按照以下步骤进行初始化。

- **创建OBS客户端**
- **配置OBS客户端**
- **配置SDK日志**

4.2 创建 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS客户端（ObsClient）是访问OBS服务的Android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 Android SDK向OBS发起请求，您需要初始化一个ObsClient实例，并根据需要修改ObsConfiguration的默认配置项。

- 直接使用服务地址创建OBS客户端（ObsClient），所有配置均为默认值，且后续不支持修改。示例代码如下：

- 永久访问密钥（AK/SK）创建OBS客户端的代码如下:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 使用访问OBS

// 关闭obsClient
obsClient.close();
```

- 临时访问密钥（AK/SK/SecurityToken）创建OBS客户端的代码如下:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String securityToken = "*** Provide your Security Token ***";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, securityToken, endPoint);

// 使用访问OBS

// 关闭obsClient
obsClient.close();
```

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。

- 使用可定制各参数的配置类（ObsConfiguration）创建OBS客户端（ObsClient），创建完成后不支持再次修改参数，具体可配置的参数参见**配置OBS客户端**。示例代码如下:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsConfiguration配置类实例
ObsConfiguration config = new ObsConfiguration();
config.setEndPoint(endPoint);
config.setSocketTimeout(30000);
config.setMaxErrorRetry(1);

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, config);

// 使用访问OBS

// 关闭obsClient
obsClient.close();
```


说明

- 您的工程中可以有多个ObsClient，也可以只有一个ObsClient。
- ObsClient是线程安全的，可在并发场景下使用。

须知

ObsClient在调用ObsClient.close方法关闭后不能再次使用。

4.3 配置 OBS 客户端

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当使用配置类（ObsConfiguration）创建OBS客户端（ObsClient）时，您可通过ObsConfiguration配置类对ObsClient进行配置，可配置代理、连接超时、最大连接数等参数。通过ObsConfiguration可以设置的参数见下表：

参数	描述	方法	建议值
connectionTimeout	建立HTTP/HTTPS连接的超时时间（单位：毫秒）。默认为60000毫秒。	ObsConfiguration.setConnectionTimeout	[10000, 60000]
socketTimeout	Socket层传输数据的超时时间（单位：毫秒）。默认为60000毫秒。	ObsConfiguration.setSocketTimeout	[10000, 60000]
idleConnectionTime	如果空闲时间超过此参数的设定值，则关闭连接（单位：毫秒）。默认为30000毫秒。	ObsConfiguration.setIdleConnectionTime	默认
maxIdleConnections	连接池中最大空闲连接数，默认值：1000。	ObsConfiguration.setMaxIdleConnections	N/A
maxConnections	最大允许的HTTP并发请求数。默认为1000。	ObsConfiguration.setMaxConnections	默认

参数	描述	方法	建议值
maxErrorRetry	请求失败（请求异常、服务端报500或503错误等）后最大的重试次数。默认3次。 **说明** 该参数对于上传对象和下载对象接口时，当上传或下载已经进入数据流处理阶段后产生异常中断，此时将不会重试。	ObsConfiguration.setMaxErrorRetry	[0, 5]
endPoint	连接OBS的服务地址。可包含协议类型、域名、端口号。示例：https://your-endpoint:443。 （出于安全性考虑，建议使用https协议）	ObsConfiguration.setEndPoint	N/A
httpProxy	HTTP代理配置。默认为空。	ObsConfiguration.setHttpProxy	N/A
validateCertificate	是否验证服务端证书。默认为false。	ObsConfiguration.setValidateCertificate	N/A
verifyResponseContentType	是否验证响应头信息的ContentType。默认为true。	ObsConfiguration.setVerifyResponseContentType	默认
uploadStreamRetryBufferSize	上传流对象时使用的缓存大小（单位：字节）。默认为512KB。	ObsConfiguration.setUploadStreamRetryBufferSize	N/A
readBufferSize	从Socket流下载对象的缓存大小（单位：字节），-1表示不设置缓存。默认为-1。	ObsConfiguration.setReadBufferSize	N/A
writeBufferSize	上传对象到Socket流时的缓存大小（单位：字节），-1表示不设置缓存。默认为-1。	ObsConfiguration.setWriteBufferSize	N/A

参数	描述	方法	建议值
socketWriteBufferSize	Socket发送缓冲区大小（单位：字节），对应java.net.SocketOptions.SO_SNDBUF参数。默认为-1表示不设置。	ObsConfiguration.setSocketWriteBufferSize	默认
socketReadBufferSize	Socket接收缓冲区大小（单位：字节），对应java.net.SocketOptions.SO_RCVBUF参数。默认为-1表示不设置。	ObsConfiguration.setSocketReadBufferSize	默认
keyManagerFactory	用于生成javax.net.ssl.KeyManager的工厂。默认为空。	ObsConfiguration.setKeyManagerFactory	N/A
trustManagerFactory	用于生成javax.net.ssl.TrustManager的工厂。默认为空。	ObsConfiguration.setTrustManagerFactory	N/A
isStrictHostnameVerification	是否严格验证服务端主机名。默认为false。	ObsConfiguration.setIsStrictHostnameVerification	N/A
keepAlive	是否使用长连接访问OBS服务。默认为true。	ObsConfiguration.setKeepAlive	N/A
cname	是否通过自定义域名访问OBS服务。默认为false。	ObsConfiguration.setCname	N/A
sslProvider	SSLContext的Provider，默认使用JDK提供的SSLContext。	ObsConfiguration.setSslProvider	N/A
httpProtocolType	访问OBS服务端时使用的HTTP协议类型。默认为HTTP1.1协议。	ObsConfiguration.setHttpProtocolType	N/A
httpDispatcher	自定义分发器。	ObsConfiguration.setHttpDispatcher	N/A

说明

- 建议值为N/A的表示需要根据实际情况进行设置。
- 如需提高大文件上传下载性能，在网络带宽满足的情况下，可对socketWriteBufferSize，sockeReadBufferSize，readBufferSize，writeBufferSize四个参数进行调优。
- 如网络状况不佳，建议增大connectionTimeout和socketTimeout的值。
- 如果设置的endPoint不带协议类型，则默认使用HTTPS协议。
- 出于DNS解析性能和OBS服务可靠性的考虑，不允许将endPoint设置为IP，必须使用域名访问OBS服务。

4.4 配置 SDK 日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS Android SDK基于java.util.logging库提供了日志功能，您可以通过LogConfigurator.enableLog开启或LogConfigurator.disableLog关闭日志功能。示例代码如下：

```
// 设置日志的级别。默认为LogConfigurator.WARN
LogConfigurator.setLogLevel(LogConfigurator.INFO);

// 设置保留日志文件的个数。默认为10
LogConfigurator.setLogFileRolloverCount(5);

// 设置每个日志文件的大小，单位：字节。默认为不限制
LogConfigurator.setLogFileSize(1024 * 1024 * 10);

// 设置日志文件存放的目录，默认存放在SD卡的logs目录下。此处以“/storage/sdcard”为例，用户可根据自身情况调整
LogConfigurator.setLogFileDir("/storage/sdcard");

// 开启日志
LogConfigurator.enableLog();

// 关闭日志
LogConfigurator.disableLog();
```

说明

- 日志功能默认是关闭的，需要主动开启。
- 您可以从**日志分析**章节获取更多关于SDK日志的介绍。
- 如果不设置日志文件存放的目录，日志文件默认存放于SD卡的logs目录下。

5 管理桶

5.1 创建桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.createBucket创建桶。

带参数创建

创建桶时可以指定桶的访问权限、存储类型和区域位置。OBS支持的桶的存储类型有三类，参见**桶存储类型**。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObsBucket obsBucket = new ObsBucket();
obsBucket.setBucketName("bucketname");
// 设置桶访问权限为公共读，默认是私有读写
obsBucket.setAcl(AccessControlList.REST_CANNED_PUBLIC_READ);
// 设置桶的存储类型为归档存储
obsBucket.setBucketStorageClass(StorageClassEnum.COLD);
// 设置桶区域位置
obsBucket.setLocation("bucketlocation");
// 创建桶
try{
    // 创建桶成功
    HeaderResponse response =obsClient.createBucket(obsBucket);
    Log.i("CreateBucket", response.getRequestId());
}
catch (ObsException e)
```

```
{
    // 创建桶失败
    Log.e("CreateBucket", "Response Code: " + e.getResponseCode());
    Log.e("CreateBucket", "Error Message: " + e.getErrorMessage());
    Log.e("CreateBucket", "Error Code:       " + e.getErrorCode());
    Log.e("CreateBucket", "Request ID:      " + e.getErrorRequestId());
    Log.e("CreateBucket", "Host ID:           " + e.getErrorHostId());
}
```

简单创建

以下代码展示如何新建一个桶：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// 创建桶
try{
    // 创建桶成功
    HeaderResponse response = obsClient.createBucket("bucketname");
    Log.i("CreateBucket", response.getRequestId());
}
catch (ObsException e)
{
    // 创建桶失败
    Log.e("CreateBucket", "Response Code: " + e.getResponseCode());
    Log.e("CreateBucket", "Error Message: " + e.getErrorMessage());
    Log.e("CreateBucket", "Error Code:       " + e.getErrorCode());
    Log.e("CreateBucket", "Request ID:      " + e.getErrorRequestId());
    Log.e("CreateBucket", "Host ID:           " + e.getErrorHostId());
}
```

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、“-”、“.”。
  - 禁止使用类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。
- 本示例创建的桶的**访问权限**默认是私有读写，存储类型默认是标准类型，区域位置是默认区域。

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。
- 您可以使用**带参数创建**方式，在创建桶时，指定桶的区域位置。

5.2 列举桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.listBuckets列举桶。以下代码展示如何获取桶列表：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 列举桶
ListBucketsRequest request = new ListBucketsRequest();
request.setQueryLocation(true);
List<ObsBucket> buckets = obsClient.listBuckets(request);
for(ObsBucket bucket : buckets){
    Log.i("ListBuckets", "BucketName" + bucket.getBucketName());
    Log.i("ListBuckets", "CreationDate" + bucket.getCreationDate());
    Log.i("ListBuckets", "Location:" + bucket.getLocation());
}
```

说明

- 获取到的桶列表将按照桶名字典顺序排列。
- 设置ListBucketsRequest.setQueryLocation参数在列举桶时查询桶的区域位置。

5.3 删除桶

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.deleteBucket删除桶。以下代码展示如何删除一个桶：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
```

```
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 删除桶
obsClient.deleteBucket("bucketname");
```

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。

5.4 判断桶是否存在

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.headBucket接口判断该桶是否已存在。

本示例用于判断桶名为“bucketname”是否存在。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

boolean exists = obsClient.headBucket("bucketname");
```

说明

- 如果抛出异常且HTTP状态码为404，表明桶不存在。

5.5 获取桶元数据

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketMetadata接口获取桶元数据。

本示例用于获取桶名为“bucketname”的元数据信息。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketMetadataInfoRequest request = new BucketMetadataInfoRequest("bucketname");
request.setOrigin("http://www.a.com");
//获取桶元数据
BucketMetadataInfoResult result = obsClient.getBucketMetadata(request);
Log.i("GetBucketMetadata", "\t:" + result.getDefaultStorageClass());
Log.i("GetBucketMetadata", "\t:" + result.getAllowOrigin());
Log.i("GetBucketMetadata", "\t:" + result.getMaxAge());
Log.i("GetBucketMetadata", "\t:" + result.getAllowHeaders());
Log.i("GetBucketMetadata", "\t:" + result.getAllowMethods());
Log.i("GetBucketMetadata", "\t:" + result.getExposeHeaders());
```

说明

- BucketMetadataInfoResult.getAllowMethods等方法的值参见桶的**跨域资源共享**（CORS）配置。

5.6 管理桶访问权限

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

桶访问权限（**ACL**）可以通过三种方式设置：

1. 创建桶时指定预定义访问策略。
2. 调用ObsClient.setBucketAcl指定预定义访问策略。
3. 调用ObsClient.setBucketAcl直接设置。

OBS支持的桶或对象权限包含五类，见下表：

权限	描述	OBS Android SDK对应值
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。	Permission.PERMISSION_READ
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。	Permission.PERMISSION_WRITE

权限	描述	OBS Android SDK对应值
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。	Permission.PERMISSION_READ_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。	Permission.PERMISSION_WRITE_ACP
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。	Permission.PERMISSION_FULL_CONTROL

OBS预定义的访问策略包含五类，见下表：

权限	描述	OBS Android SDK对应值
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。	AccessControlList.REST_CANNED_PRIVATE
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、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。	AccessControlList.REST_CANNED_PUBLIC_READ
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、拷贝段、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。	AccessControlList.REST_CANNED_PUBLIC_READ_WRITE

权限	描述	OBS Android SDK对应值
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。	AccessControlList.REST_CANNED_PUBLIC_READ_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、拷贝段、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。	AccessControlList.REST_CANNED_PUBLIC_READ_WRITE_DELIVERED

创桶时指定预定义访问策略

以下代码展示如何在创建桶时指定预定义访问策略：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObsBucket obsBucket = new ObsBucket();
obsBucket.setBucketName("bucketname");
// 设置桶访问权限为公共读写
obsBucket.setAcl(AccessControlList.REST_CANNED_PUBLIC_READ_WRITE);
// 创建桶
obsClient.createBucket(obsBucket);
```

为桶设置预定义访问策略

以下代码展示如何为桶设置预定义访问策略：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 设置桶访问权限为私有读写
obsClient.setBucketAcl("bucketname", AccessControlList.REST_CANNED_PRIVATE);
```

直接设置桶访问权限

以下代码展示如何直接设置桶访问权限：


```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
owner.setId("ownerid");
acl.setOwner(owner);
// 为指定用户设置完全控制权限
acl.grantPermission(new CanonicalGrantee("userid"), Permission.PERMISSION_FULL_CONTROL);
// 为所有用户设置读权限
acl.grantPermission(GroupGrantee.ALL_USERS, Permission.PERMISSION_READ);
// 直接设置桶访问权限
obsClient.setBucketAcl("bucketname", acl);
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取桶访问权限

您可以通过ObsClient.getBucketAcl获取桶的访问权限。以下代码展示如何获取桶访问权限：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AccessControlList acl = obsClient.getBucketAcl("bucketname");
Log.i("GetBucketAcl", acl.toString());
```

5.7 管理桶策略

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

除了桶访问权限外，桶的拥有者还可以通过桶策略，提供对桶和桶内对象的集中访问控制。

更多关于桶策略的内容请参考**桶策略**。

设置桶策略

您可以通过ObsClient.setBucketPolicy设置桶策略。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
obsClient.setBucketPolicy("bucketname", "your policy");
```

说明

桶策略内容的具体格式（JSON格式字符串）请参考《对象存储服务API参考》。

获取桶策略

您可以通过ObsClient.getBucketPolicy获取桶策略。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

String policy = obsClient.getBucketPolicy("bucketname");
Log.i("GetBucketPolicy","\t" + policy);
```

删除桶策略

您可以通过ObsClient.deleteBucketPolicy删除桶策略。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

obsClient.deleteBucketPolicy("bucketname");
```

5.8 获取桶区域位置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketLocation获取桶的区域位置。

本示例用于获取桶名为“bucketname”的区域位置信息。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

String location = obsClient.getBucketLocation("bucketname");
Log.i("GetBucketLocation", "\t:" + location);
```

说明

- 创建桶时可以指定桶的区域位置，请参见**创建桶**。

5.9 获取桶存量信息

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

桶存量信息包括桶已使用的空间大小以及桶包含的对象个数。

您可以通过ObsClient.getBucketStorageInfo获取桶的存量信息。

本示例用于获取桶名为“bucketname”的存量信息。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketStorageInfo storageInfo = obsClient.getBucketStorageInfo("bucketname");
Log.i("GetBucketStorageInfo", "\t" + storageInfo.getObjectNumber());
Log.i("GetBucketStorageInfo", "\t" + storageInfo.getSize());
```

说明

- 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服务端错误码**。

5.10 桶配额

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

设置桶配额

您可以通过ObsClient.setBucketQuota设置桶配额。以下代码展示如何设置桶配额：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 设置桶配额为100MB
BucketQuota quota = new BucketQuota(1024 * 1024 * 100);
obsClient.setBucketQuota("bucketname", quota);
```

说明

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。

获取桶配额

您可以通过ObsClient.getBucketQuota获取桶配额。以下代码展示如何获取桶配额：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketQuota quota = obsClient.getBucketQuota("bucketname");
Log.i("GetBucketQuota", "\t" + quota.getBucketQuota());
```

5.11 桶存储类型

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS允许您对桶配置不同的存储类型，桶中对象的存储类型默认将与桶的存储类型保持一致。不同的存储类型可以满足客户业务对存储性能、成本的不同诉求。桶的存储类型分为三类，见下表：

类型	说明	OBS Android SDK对应值
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。	StorageClassEnum.STANDARD
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。	StorageClassEnum.WARM
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。	StorageClassEnum.COLD

更多关于桶存储类型的内容请参考**桶的存储类型**。

设置桶存储类型

您可以通过ObsClient.setBucketStoragePolicy设置桶存储类型。以下代码展示如何设置桶存储类型：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 设置桶的存储类型为低频访问存储.
BucketStoragePolicyConfiguration storgePolicy = new BucketStoragePolicyConfiguration();
storgePolicy.setBucketStorageClass(StorageClassEnum.WARM);
obsClient.setBucketStoragePolicy("bucketname", storgePolicy);
```

获取桶存储类型

您可以通过ObsClient.getBucketStoragePolicy获取桶存储类型。以下代码展示如何获取桶存储类型：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```



```
BucketStoragePolicyConfiguration storagePolicy = obsClient.getBucketStoragePolicy("bucketname");
Log.i("GetBucketStoragePolicy", "\t" + storagePolicy.getBucketStorageClass());
```


6 上传对象

6.1 对象上传简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

在OBS中，用户操作的基本数据单元是对象。OBS Android SDK提供了丰富的对象上传接口，可以通过以下方式上传对象：

- **流式上传**
- **文件上传**
- **分段上传**
- **追加上传**
- **断点续传上传**
- **基于表单上传**

SDK支持上传0KB~5GB的对象。流式上传、文件上传和追加上传的内容大小不能超过5GB；当上传较大文件时，请使用分段上传，分段上传每段内容大小不能超过5GB；基于表单上传提供了基于浏览器表单上传对象的方式。

如果上传的对象权限设置为匿名用户读取权限，对象上传成功后，匿名用户可通过链接地址访问该对象数据。对象链接地址格式为：https://*桶名.域名/文件夹目录层级/对象名*。如果该对象存在于桶的根目录下，则链接地址将不需要有文件夹目录层级。

说明

以下示例代码中的"bucketname"为待上传对象的目标桶的名称；"objectname"为上传后，期望在桶内生成的目标对象名称，可指定文件夹目录层级，例如src/src1/src2/test.txt。如果不指定目录层级，则该对象上传到桶的根目录下。

6.2 流式上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

流式上传使用java.io.InputStream作为对象的数据源。您可以通过ObsClient.putObject上传您的数据流到OBS。以下代码展示了如何进行流式上传：

上传字符串（byte 数组）

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

String content = "Hello OBS";
obsClient.putObject("bucketname", "objectname", new ByteArrayInputStream(content.getBytes()));
```

上传网络流

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

InputStream inputStream = new URL("http://www.a.com").openStream();
obsClient.putObject("bucketname", "objectname", inputStream);
```

上传文件流

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

FileInputStream fis = new FileInputStream(new File("localfile")); // localfile为待上传的本地文件路径，需要指定到具体的文件名
obsClient.putObject("bucketname", "objectname", fis);
// 待上传的本地文件路径，需要指定到具体的文件名
FileInputStream fis2 = new FileInputStream(new File("localfile2"));
```

```
PutObjectRequest request = new PutObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("objectname2");
request.setInput(fis2);
obsClient.putObject(request);
```

说明

- 推荐使用**文件上传**的形式上传本地文件，而不是文件流形式。
- 大文件上传建议使用**分段上传**。
- 上传的内容大小不能超过5GB。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 PutObjectRequest.setIsEncodeHeaders(false) 关闭自动编码。

6.3 文件上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

文件上传使用本地文件作为对象的数据源。以下代码展示了如何进行文件上传：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
try {
    // 文件上传
    // localfile 为待上传的本地文件路径，需要指定到具体的文件名
    PutObjectRequest request = new PutObjectRequest();
    request.setBucketName("examplebucket");
    request.setObjectKey("objectkey");
    request.setFile(new File("localfile"));
    obsClient.putObject(request);
    System.out.println("putObject successfully");
} catch (ObsException e) {
    System.out.println("putObject failed");
    // 请求失败,打印http状态码
    System.out.println("HTTP Code:" + e.getResponseCode());
    // 请求失败,打印服务端错误码
    System.out.println("Error Code:" + e.getErrorCode());
    // 请求失败,打印详细错误信息
    System.out.println("Error Message:" + e.getErrorMessage());
    // 请求失败,打印请求id
    System.out.println("Request ID:" + e.getErrorRequestId());
    System.out.println("Host ID:" + e.getErrorHostId());
    e.printStackTrace();
} catch (Exception e) {
    System.out.println("putObject failed");
```



```
    // 其他异常信息打印
    e.printStackTrace();
}
```

说明

- 上传内容大小不能超过5GB。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 PutObjectRequest.setIsEncodeHeaders(false) 关闭自动编码。如需使用该功能，请安装最新版本的SDK。

6.4 获取上传进度

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过PutObjectRequest.setProgressListener设置数据传输接口来获取上传的进度。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

PutObjectRequest request = new PutObjectRequest("bucketname", "objectname");
request.setFile(new File("localfile")); // localfile为上传的本地文件路径，需要指定到具体的文件名
request.setProgressListener(new ProgressListener() {

    @Override
    public void progressChanged(ProgressStatus status) {
        // 获取上传平均速率
        Log.i("PutObject", "AverageSpeed:" + status.getAverageSpeed());
        // 获取上传进度百分比
        Log.i("PutObject", "TransferPercentage:" + status.getTransferPercentage());
    }
});
// 每上传1MB数据反馈上传进度
request.setProgressInterval(1024 * 1024L);
obsClient.putObject(request);
```

说明

- 支持获取上传进度的接口包括：流式上传、文件上传、上传段、追加上传和断点续传上传；
- 如果ProgressStatus.getTransferPercentage()返回-1，表明是流式上传，此时必须设置对象属性中的对象长度（Content-Length）。

6.5 创建文件夹

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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。

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

final String keySuffixWithSlash = "parent_directory/";
obsClient.putObject("bucketname", keySuffixWithSlash, new ByteArrayInputStream(new byte[0]));

// 在文件夹下创建对象
obsClient.putObject("bucketname", keySuffixWithSlash + "objectname", new ByteArrayInputStream("Hello OBS".getBytes()));
```

说明

- 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/。

6.6 设置对象属性

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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。

对象属性详细说明见下表：

名称	描述	默认值
对象长度（Content-Length）	上传对象的长度，超过流/文件的长度会截断。	流/文件实际长度
对象MIME类型（Content-Type）	对象的MIME类型，定义对象的类型及网页编码，决定浏览器将以什么形式、什么编码读取对象。	application/octet-stream
对象MD5值（Content-MD5）	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。	无
对象存储类型	对象的存储类型，不同的存储类型可以满足客户业务对存储性能、成本的不同诉求。默认与桶的存储类型保持一致，可以设置为与桶的存储类型不同。	无
对象自定义元数据	用户对上传到桶中对象的自定义属性描述，以便对对象进行自定义管理。	无

设置对象长度

您可以通过ObjectMetadata.setContentLength来设置对象长度。以下代码展示如何设置对象长度：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentLength(1024 * 1024L);//1MB
obsClient.putObject("bucketname", "objectname", new File("localfile"), metadata);
```

设置对象 MIME 类型

您可以通过ObjectMetadata.setContentType来设置对象MIME类型。以下代码展示如何设置对象MIME类型：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
```



```
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 上传图片
ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("image/jpeg");
obsClient.putObject("bucketname", "objectname", new File("localimage.jpg"), metadata);
```

说明

如果不设置对象MIME类型，SDK会根据上传文件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。

设置对象 MD5 值

您可以通过ObjectMetadata.setContentMd5来设置对象MD5值。以下代码展示如何设置对象MD5值：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// 上传图片
ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentMd5("your md5 which should be encoded by base64");
obsClient.putObject("bucketname", "objectname", new File("localimage.jpg"), metadata);
```

说明

- 对象数据的MD5值必须经过Base64编码。
- 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。
- 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。
- 您可以通过ObsClient.base64Md5来直接计算Content-MD5头域。

设置对象存储类型

您可以通过ObjectMetadata.setStorageClass来设置对象存储类型。以下代码展示如何设置对象存储类型：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObjectMetadata metadata = new ObjectMetadata();
// 设置对象存储类型为低频访问存储
metadata.setObjectStorageClass(StorageClassEnum.WARM);
obsClient.putObject("bucketname", "objectname", new File("localfile"), metadata);
```


说明

- 如果不设置，对象的存储类型默认与桶的存储类型保持一致。
- 对象的存储类型分为三类，其含义与**桶存储类型**一致。
- 下载归档存储类型的对象前必须将其恢复。

设置对象自定义元数据

您可以通过ObjectMetadata.addUserMetadata来设置对象自定义元数据。以下代码展示如何设置对象自定义元数据：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObjectMetadata metadata = new ObjectMetadata();
metadata.addUserMetadata("property1", "property-value1");
metadata.getMetadata().put("property2", "property-value2");
obsClient.putObject("bucketname", "objectname", new File("localfile"), metadata);
```

说明

- 在上面设置对象自定义元数据示例代码中，用户自定义了一个名称为“property1”，值为“property-value1”的元数据和一个名称为“property2”，值为“property-value2”的元数据。
- 一个对象可以有多个元数据，总大小不能超过8KB。
- 对象的自定义元数据可以通过ObsClient.getObjectMetadata获取，参见**获取对象元数据**。
- 使用ObsClient.getObject下载对象时，对象的自定义元数据也会同时下载。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为 ”attachment; filename="中文.txt"”，则对象元数据中存储的信息为”attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 PutObjectRequest.setIsEncodeHeaders(false) 关闭自动编码。

6.7 分段上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

对于较大文件上传，可以切分成段上传。用户可以在如下的应用场景内（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服务端之间的链接经常断开。

- 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：

步骤1 初始化分段上传任务（ObsClient.initiateMultipartUpload）。

步骤2 逐个或并行上传段（ObsClient.uploadPart）。

步骤3 合并段（ObsClient.completeMultipartUpload）或取消分段上传任务(ObsClient.abortMultipartUpload)。

----结束

初始化分段上传任务

使用分段上传方式传输数据前，必须先通知OBS初始化一个分段上传任务。该操作会返回一个OBS服务端创建的全局唯一标识（Upload ID），用于标识本次分段上传任务。您可以根据这个唯一标识来发起相关的操作，如取消分段上传任务、列举分段上传任务、列举已上传的段等。

您可以通过ObsClient.initiateMultipartUpload初始化一个分段上传任务：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest("bucketname", "objectname");
ObjectMetadata metadata = new ObjectMetadata();
metadata.addUserMetadata("property", "property-value");
metadata.setContentType("text/plain");
request.setMetadata(metadata);
InitiateMultipartUploadResult result = obsClient.initiateMultipartUpload(request);

String uploadId = result.getUploadId();
Log.i("InitiateMultipartUpload", "\t" + uploadId);
```

说明

- 用InitiateMultipartUploadRequest指定上传对象的名称和所属桶。
- 在InitiateMultipartUploadRequest中，您可以设置对象MIME类型、对象存储类型、对象自定义元数据等对象属性。
- InitiateMultipartUploadResult.getUploadId返回分段上传任务的全局唯一标识（Upload ID），在后面的操作中将用到它。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 InitiateMultipartUploadRequest.setIsEncodeHeaders(false) 关闭自动编码。

上传段

初始化一个分段上传任务之后，可以根据指定的对象名和Upload ID来分段上传数据。每一个上传的段都有一个标识它的号码——分段号（Part Number，范围是

1~10000）。对于同一个Upload ID，该分段号不但唯一标识这一段数据，也标识了这段数据在整个对象内的相对位置。如果您用同一个分段号上传了新的数据，那么OBS上已有的这个段号的数据将被覆盖。除了最后一段以外，其他段的大小范围是100KB~5GB；最后段大小范围是0~5GB。每个段不需要按顺序上传，甚至可以在不同进程、不同机器上上传，OBS会按照分段号排序组成最终对象。

您可以通过ObsClient.uploadPart上传段：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

List<PartEtag> partEtags = new ArrayList<PartEtag>();
// 上传第一段
UploadPartRequest request = new UploadPartRequest("bucketname", "objectname");
// 设置Upload ID
request.setUploadId(uploadId);
// 设置分段号，范围是1~10000，
request.setPartNumber(1);
// 设置将要上传的大文件，其中localfile为待上传的本地文件路径，需要指定到具体的文件名
request.setFile(new File("localfile"));

// 设置分段大小
request.setPartSize(5 * 1024 * 1024L);
UploadPartResult result = obsClient.uploadPart(request);
partEtags.add(new PartEtag(result.getEtag(), result.getPartNumber()));

// 上传第二段
request = new UploadPartRequest("bucketname", "objectname");
// 设置Upload ID
request.setUploadId(uploadId);
// 设置分段号
request.setPartNumber(2);
// 设置将要上传的大文件
request.setFile(new File("localfile"));
// 设置第二段的段偏移量
request.setOffset(5 * 1024 * 1024L);
// 设置分段大小
request.setPartSize(5 * 1024 * 1024L);
result = obsClient.uploadPart(request);
partEtags.add(new PartEtag(result.getEtag(), result.getPartNumber()));
```

说明

- 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UploadPartRequest.setAttachMd5为true来让SDK自动计算每段数据的MD5值（仅在数据源为本地文件时有效），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。
- 可以通过UploadPartRequest.setContentMd5直接设置上传数据的MD5值，提供给OBS服务端用于校验数据完整性。如果设置了该值，UploadPartRequest.setAttachMd5参数会被忽略。
- 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。

合并段

所有分段上传完成后，需要调用合并段接口来在OBS服务端生成最终对象。在执行该操作时，需要提供所有有效的分段列表（包括分段号和分段ETag值）；OBS收到提交的分段列表后，会逐一验证每个段的有效性。当所有段验证通过后，OBS将把这些分段组合成最终的对象。

您可以通过ObsClient.completeMultipartUpload合并段：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

List<PartEtag> partEtags = new ArrayList<PartEtag>();

// 第一段
PartEtag part1 = new PartEtag();
part1.setPartNumber(1);
part1.seteTag("etag1");
partEtags.add(part1);

// 第二段
PartEtag part2 = new PartEtag();
part2.setPartNumber(2);
part2.seteTag("etag2");
partEtags.add(part2);

CompleteMultipartUploadRequest request = new CompleteMultipartUploadRequest("bucketname", "objectname", uploadId, partEtags);

obsClient.completeMultipartUpload(request);
```

说明

- 上面代码中的 partEtags是进行上传段后保存的分段号和分段ETag值的列表，它必须是按分段号升序排列。
- 分段可以是不连续的。

并发分段上传

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。下面的示例代码展示了如何使用分段上传并发上传大文件：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
final String bucketName = "bucketname";
final String objectKey = "objectname";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 初始化线程池
```

```
ExecutorService executorService = Executors.newFixedThreadPool(20);
final File largeFile = new File("localfile");

// 初始化分段上传任务
InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest(bucketName, objectKey);
InitiateMultipartUploadResult result = obsClient.initiateMultipartUpload(request);

final String uploadId = result.getUploadId();
Log.i("UploadPart", "\t"+ uploadId + "\n");

// 每段上传100MB
long partSize = 100 * 1024 * 1024L;
long fileSize = largeFile.length();

// 计算需要上传的段数
long partCount = fileSize % partSize == 0 ? fileSize / partSize : fileSize / partSize + 1;

final List<PartEtag> partEtags = Collections.synchronizedList(new ArrayList<PartEtag>());

// 执行并发上传段
for (int i = 0; i < partCount; i++)
{
    // 分段在文件中的起始位置
    final long offset = i * partSize;
    // 分段大小
    final long currPartSize = (i + 1 == partCount) ? fileSize - offset : partSize;
    // 分段号
    final int partNumber = i + 1;
    executorService.execute(new Runnable()
    {
        @Override
        public void run()
        {
            UploadPartRequest uploadPartRequest = new UploadPartRequest();
            uploadPartRequest.setBucketName(bucketName);
            uploadPartRequest.setObjectKey(objectKey);
            uploadPartRequest.setUploadId(uploadId);
            uploadPartRequest.setFile(largeFile);
            uploadPartRequest.setPartSize(currPartSize);
            uploadPartRequest.setOffset(offset);
            uploadPartRequest.setPartNumber(partNumber);

            UploadPartResult uploadPartResult;
            try
            {
                uploadPartResult = obsClient.uploadPart(uploadPartRequest);
                Log.i("UploadPart", "Part#" + partNumber + " done\n");
                partEtags.add(new PartEtag(uploadPartResult.getEtag(), uploadPartResult.getPartNumber()));
            }
            catch (ObsException e)
            {
                Log.e("UploadPart", e.getMessage(), e);
            }
        }
    });
}

// 等待上传完成
executorService.shutdown();
while (!executorService.isTerminated())
{
    try
    {
        executorService.awaitTermination(5, TimeUnit.SECONDS);
    }
    catch (InterruptedException e)
    {
        Log.e("UploadPart", e.getMessage(), e);
    }
```



```
}

// 合并段
CompleteMultipartUploadRequest completeMultipartUploadRequest =
    new CompleteMultipartUploadRequest(bucketName, objectKey, uploadId, partEtags);
obsClient.completeMultipartUpload(completeMultipartUploadRequest);
```

说明

大文件分段上传时，使用UploadPartRequest.setOffset和UploadPartRequest.setPartSize来设置每段的起始结束位置。

取消分段上传任务

分段上传任务可以被取消，当一个分段上传任务被取消后，就不能再使用其Upload ID做任何操作，已经上传段也会被OBS删除。

采用分段上传方式上传对象过程中或上传对象失败后会在桶内产生段，这些段会占用您的存储空间，您可以通过取消该分段上传任务来清理掉不需要的段，节约存储空间。

您可以通过ObsClient.abortMultipartUpload取消分段上传任务：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AbortMultipartUploadRequest request = new AbortMultipartUploadRequest("bucketname", "objectname", uploadId);

obsClient.abortMultipartUpload(request);
```

列举已上传的段

您可使用ObsClient.listParts列举出某一分段上传任务所有已经上传成功的段。

该接口可设置的参数如下：

参数	作用	OBS Android SDK对应方法
bucketName	分段上传任务所属的桶名。	ListPartsRequest.setBucketName
key	分段上传任务所属的对象名。	ListPartsRequest.setKey
uploadId	分段上传任务全局唯一标识，从ObsClient.initiateMultipartUpload返回的结果获取。	ListPartsRequest.setUploadId

参数	作用	OBS Android SDK对应方法
maxParts	表示列举已上传的段返回结果最大段数目，即分页时每一页中段数目。	ListPartsRequest.setMaxParts
partNumberMarker	表示待列出段的起始位置，只有Part Number大于该参数的段会被列出。	ListPartsRequest.setPartNumberMarker

- 简单列举

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

//列举已上传的段，其中uploadId来自于initiateMultipartUpload
ListPartsRequest request = new ListPartsRequest("bucketname", "objectname");
request.setUploadId(uploadId);
ListPartsResult result = obsClient.listParts(request);

for(Multipart part : result.getMultipartList()){
    // 分段号，上传时候指定
   Log.i("ListParts", "\t"+part.getPartNumber());
    // 段数据大小
   Log.i("ListParts","\t"+part.getSize());
    // 分段的ETag值
   Log.i("ListParts","\t"+part.getEtag());
    // 段的最后上传时间
   Log.i("ListParts","\t"+part.getLastModified());
}
```

说明

- 列举段至多返回1000个段信息，如果指定的Upload ID包含的段数量大于1000，则返回结果中ListPartsResult.isTruncated为true表明本次没有返回全部段，并可通过ListPartsResult.getNextPartNumberMarker获取下次列举的起始位置。
- 如果想获取指定Upload ID包含的所有分段，可以采用分页列举的方式。

- 列举所有段

由于ObsClient.listParts只能列举至多1000个段，如果段数量大于1000，列举所有分段请参考如下示例：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
```



```
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 列举所有已上传的段，其中uploadId来自于initiateMultipartUpload
ListPartsRequest request = new ListPartsRequest("bucketname", "objectname");
request.setUploadId(uploadId);
ListPartsResult result;

do{
    result = obsClient.listParts(request);
    for(Multipart part : result.getMultipartList()){
        // 分段号，上传时候指定
        Log.i("ListParts","\t"+part.getPartNumber());
        // 段数据大小
        Log.i("ListParts","\t"+part.getSize());
        // 分段的ETag值
        Log.i("ListParts","\t"+part.getEtag());
        // 段的最后上传时间
        Log.i("ListParts","\t"+part.getLastModified());
    }
    request.setPartNumberMarker(Integer.parseInt(result.getNextPartNumberMarker()));
}while(result.isTruncated());
```

列举分段上传任务

您可以通过ObsClient.listMultipartUploads列举分段上传任务。列举分段上传任务可设置的参数如下：

参数	作用	OBS Android SDK对应方法
bucketName	桶名。	ListMultipartUploadsRequest.setBucketName
prefix	限定返回的分段上传任务中的对象名必须带有prefix前缀。	ListMultipartUploadsRequest.setPrefix
delimiter	用于对分段上传任务中的对象名进行分组的字符。对于对象名中包含delimiter的任务，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。	ListMultipartUploadsRequest.setDelimiter
maxUploads	列举分段上传任务的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。	ListMultipartUploadsRequest.setMaxUploads
keyMarker	表示列举时返回指定的keyMarker之后的分段上传任务。	ListMultipartUploadsRequest.setKeyMarker
uploadIdMarker	只有与keyMarker参数一起使用时才有意义，用于指定返回结果的起始位置，即列举时返回指定keyMarker的uploadIdMarker之后的分段上传任务。	ListMultipartUploadsRequest.setUploadIdMarker

- 简单列举分段上传任务

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListMultipartUploadsRequest request = new ListMultipartUploadsRequest("bucketname");

MultipartUploadListing result = obsClient.listMultipartUploads(request);
for(MultipartUpload upload : result.getMultipartTaskList()){
    Log.i("ListMultipartUploads","\t" + upload.getUploadId());
    Log.i("ListMultipartUploads","\t" + upload.getObjectKey());
    Log.i("ListMultipartUploads","\t" + upload.getInitiatedDate());
}
```

说明

- 列举分段上传任务至多返回1000个任务信息，如果指定的桶包含的分段上传任务数量大于1000，则MultipartUploadListing.isTruncated为true表明本次没有返回全部结果，并可通过MultipartUploadListing.getNextKeyMarker和MultipartUploadListing.getNextUploadIdMarker获取下次列举的起点。
- 如果想获取指定桶包含的所有分段上传任务，可以采用分页列举的方式。

- 分页列举全部分段上传任务

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
String uploadId = "upload id from initiateMultipartUpload";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListMultipartUploadsRequest request = new ListMultipartUploadsRequest("bucketname");
MultipartUploadListing result;

do{
    result = obsClient.listMultipartUploads(request);
    for(MultipartUpload upload : result.getMultipartTaskList()){
        Log.i("ListMultipartUploads","\t" + upload.getUploadId());
        Log.i("ListMultipartUploads","\t" + upload.getObjectKey());
        Log.i("ListMultipartUploads","\t" + upload.getInitiatedDate());
    }
    request.setKeyMarker(result.getNextKeyMarker());
    request.setUploadIdMarker(result.getNextUploadIdMarker());
}while(result.isTruncated());
```


6.8 设置对象生命周期

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

上传对象或者初始化分段上传任务时，您可以直接指定对象的过期时间。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
 try {
     PutObjectRequest request = new PutObjectRequest();
     request.setBucketName("examplebucket");
     request.setObjectKey("objectname");
     request.setFile(new File("localfile"));
     // 上传对象时，设置对象30天后过期
     request.setExpires(30);
     obsClient.putObject(request);
     // InitiateMultipartUploadRequest request2 = new InitiateMultipartUploadRequest("bucketname", "objectname");
     //// 初始化分段上传任务时，设置合并段后生成的对象60天后过期
     // request2.setExpires(60);
     //obsClient.initiateMultipartUpload(request);
     System.out.println("putObject successfully");
  } catch (ObsException e) {
     System.out.println("putObject failed");
     // 请求失败,打印http状态码
     System.out.println("HTTP Code:" + e.getResponseCode());
     // 请求失败,打印服务端错误码
     System.out.println("Error Code:" + e.getErrorCode());
     // 请求失败,打印详细错误信息
     System.out.println("Error Message:" + e.getErrorMessage());
     // 请求失败,打印请求id
     System.out.println("Request ID:" + e.getErrorRequestId());
     System.out.println("Host ID:" + e.getErrorHostId());
     e.printStackTrace();
 } catch (Exception e) {
     System.out.println("putObject failed");
     // 其他异常信息打印
     e.printStackTrace();
 }
```

说明

- 该方式仅支持设置以天为单位的对象过期时间，过期后的对象会被OBS服务端自动清理。
- 该方式设置的对象过期时间，其优先级高于桶生命周期规则。

6.9 追加上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

追加上传可实现对同一个对象追加数据内容的功能。您可以通过ObsClient.appendObject进行追加上传。示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 第一次追加上传
AppendObjectRequest request = new AppendObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("objectname");
request.setPosition(0L);
request.setInput(new ByteArrayInputStream("Hello OBS".getBytes()));
AppendObjectResult result = obsClient.appendObject(request);

// 第二次追加上传
request.setPosition(result.getNextPosition());
request.setInput(new ByteArrayInputStream("Hello OBS Again".getBytes()));
result = obsClient.appendObject(request);

Log.i("AppendObject", "NextPosition:" + result.getNextPosition());
Log.i("AppendObject", "Etag:" + result.getEtag());
// 通过获取对象属性接口获取下次追加上传的位置
ObjectMetadata metadata = obsClient.getObjectMetadata("bucketname", "objectname");
Log.i("AppendObject", "NextPosition from metadata:" + metadata.getNextPosition());
```

说明

- ObsClient.putObject上传的对象可覆盖ObsClient.appendObject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是追加数据内容的ETag，不是完整对象的ETag。
- 单次追加上传的内容不能超过5GB，且最多支持10000次追加上传。
- 追加上传成功后，可通过AppendObjectResult.getNextPosition获取下次追加上传的位置；或者通过ObsClient.getObjectMetadata接口获取下次追加上传的位置。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 AppendObjectRequest.setIsEncodeHeaders(false) 关闭自动编码。

6.10 分段复制

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

分段复制是分段上传的一种特殊情况，即分段上传任务中的段通过复制OBS指定桶中现有对象（或对象的一部分）来实现。您可以通过ObsClient.copyPart来复制段。

本示例用于分段复制桶名为“sourcebucketname”里的“sourceobjectname”对象到桶名为“destbucketname”里的“destobjectname”对象。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
final String destBucketName = "destbucketname";
final String destObjectKey = "destobjectname";
final String sourceBucketName = "sourcebucketname";
final String sourceObjectKey = "sourceobjectname";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 初始化线程池
ExecutorService executorService = Executors.newFixedThreadPool(20);

// 初始化分段上传任务
InitiateMultipartUploadRequest request = new InitiateMultipartUploadRequest(destBucketName, destObjectKey);
InitiateMultipartUploadResult result = obsClient.initiateMultipartUpload(request);

final String uploadId = result.getUploadId();
Log.i("CopyPart","\t"+ uploadId + "\n");

// 获取大对象信息
ObjectMetadata metadata = obsClient.getObjectMetadata(sourceBucketName, sourceObjectKey);
// 每段复制100MB
long partSize = 100 * 1024 * 1024L;
long objectSize = metadata.getContentLength();

// 计算需要复制的段数
long partCount = objectSize % partSize == 0 ? objectSize / partSize : objectSize / partSize + 1;

final List<PartEtag> partEtags = Collections.synchronizedList(new ArrayList<PartEtag>());

// 执行并发复制段
for (int i = 0; i < partCount; i++)
{
    // 复制段起始位置
    final long rangeStart = i * partSize;
    // 复制段结束位置
    final long rangeEnd = (i + 1 == partCount) ? objectSize - 1 : rangeStart + partSize - 1;
    // 分段号
    final int partNumber = i + 1;
    executorService.execute(new Runnable()
```

```
    {

        @Override
        public void run()
        {
            CopyPartRequest request = new CopyPartRequest();
            request.setUploadId(uploadId);
            request.setSourceBucketName(sourceBucketName);
            request.setSourceObjectKey(sourceObjectKey);
            request.setDestinationBucketName(destBucketName);
            request.setDestinationObjectKey(destObjectKey);
            request.setByteRangeStart(rangeStart);
            request.setByteRangeEnd(rangeEnd);
            request.setPartNumber(partNumber);
            CopyPartResult result;
            try
            {
                result = obsClient.copyPart(request);
                Log.i("CopyPart","Part#" + partNumber + " done\n");
                partEtags.add(new PartEtag(result.getEtag(), result.getPartNumber()));
            }
            catch (ObsException e)
            {
                Log.e("CopyPart", e.getMessage(), e);
            }
        }
    });
}

// 等待复制完成
executorService.shutdown();
while (!executorService.isTerminated())
{
    try
    {
        executorService.awaitTermination(5, TimeUnit.SECONDS);
    }
    catch (InterruptedException e)
    {
        Log.e("CopyPart", e.getMessage(), e);
    }
}

// 合并段
CompleteMultipartUploadRequest completeMultipartUploadRequest =
    new CompleteMultipartUploadRequest(destBucketName, destObjectKey, uploadId, partEtags);
obsClient.completeMultipartUpload(completeMultipartUploadRequest);
```

说明

复制段时，使用**PartNumber参数**指定分段号；使用**UploadId参数**指定分段上传任务的全局唯一标识；使用**CopySource参数**指定复制时的源对象信息；使用**CopySourceRange参数**指定待复制的源对象的字节范围。

6.11 断点续传上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当上传大文件时，经常出现因网络不稳定或程序崩溃导致上传失败的情况。失败后再次重新上传不仅浪费资源，而且当网络不稳定时仍然有上传失败的风险。断点续传上

传接口能有效地解决此类问题引起的上传失败，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，否则抛出异常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ObsClient.uploadFile进行断点续传上传。该接口可设置的参数如下：

参数	作用	OBS Android SDK对应方法
bucketName	桶名，必选参数。	UploadFileRequest.setBucketName
objectKey	对象名，必选参数。	UploadFileRequest.setObjectKey
uploadFile	待上传的本地文件，必选参数。	UploadFileRequest.setUploadFile
partSize	分段大小，单位字节，取值范围是100KB~5GB，默认为5MB。	UploadFileRequest.setPartSize
taskNum	分段上传时的最大并发数，默认为1。	UploadFileRequest.setTaskNum
enableCheckpoint	是否开启断点续传模式，默认为false，表示不开启。	UploadFileRequest.setEnableCheckpoint
isEncodeHeaders	是否自动编码请求头	UploadFileRequest.setIsEncodeHeaders
checkpointFile	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。	UploadFileRequest.setCheckpointFile
objectMetadata	对象的属性。	UploadFileRequest.setObjectMetadata
enableCheckSum	是否校验待上传文件的内容，只在断点续传模式下有效。默认为false，表示不校验。	UploadFileRequest.setEnableCheckSum
progressListener	设置数据传输监听器，用于获取上传进度。	UploadFileRequest.setProgressListener

以下代码展示了如何使用断点续传上传接口上传文件：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
```



```
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

UploadFileRequest request = new UploadFileRequest("bucketname", "objectname");
// 设置待上传的本地文件，其中localfile为待上传的本地文件路径，需要指定到具体的文件名
request.setUploadFile("localfile");
// 设置分段上传时的最大并发数
request.setTaskNum(5);
// 设置分段大小为10MB
request.setPartSize(10 * 1024 * 1024);
// 开启断点续传模式
request.setEnableCheckpoint(true);
try{
    // 进行断点续传上传
    CompleteMultipartUploadResult result = obsClient.uploadFile(request);
}catch (ObsException e) {
    // 发生异常时可再次调用断点续传上传接口进行重新上传
}
```

说明

- 断点续传上传接口是利用**分段上传**特性实现的，是对分段上传的封装和加强。
- 断点续传上传接口不仅能在失败重传时节省资源提高效率，还因其对分段进行并发上传的机制能加快上传速度，帮助用户快速完成上传业务；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、并发上传的实现等内部细节。
- **enableCheckpoint参数**默认是false，代表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。
- **checkpointFile参数**和**enableCheckSum参数**仅在**enableCheckpoint参数**为true时有效。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 UploadFileRequest.setIsEncodeHeaders(false) 关闭自动编码。
- 断点续传上传功能暂不支持暂停或取消操作。

6.12 基于表单上传

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

基于表单上传是使用HTML表单形式上传对象到指定桶中，对象最大不能超过5GB。

您可以通过ObsClient.createPostSignature生成基于表单上传的请求参数。使用代码模拟表单上传的完整代码示例，参见**PostObjectSample**。您也可以通过如下步骤进行表单上传：

步骤1 使用ObsClient.createPostSignature生成用于鉴权的请求参数。

步骤2 准备表单HTML页面。

步骤3 将生成的请求参数填入HTML页面。

步骤4 选择本地文件，进行表单上传。

----结束

说明

使用SDK生成用于鉴权的请求参数包括两个：

- policy，对应表单中policy字段。
- signature，对应表单中的signature字段。

以下代码展示了如何生成基于表单上传的请求参数：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

PostSignatureRequest request = new PostSignatureRequest();
// 设置表单参数
Map<String, Object> formParams = new HashMap<String, Object>();
// 设置对象访问权限为公共读
formParams.put("x-obs-acl", "public-read");
// 设置对象MIME类型
formParams.put("content-type", "text/plain");

request.setFormParams(formParams);
// 设置表单上传请求有效期，单位：秒
request.setExpires(3600);
PostSignatureResponse response = obsClient.createPostSignature(request);

// 获取表单上传请求参数
Log.i("CreatePostSignature", "\t" + response.getPolicy());
Log.i("CreatePostSignature", "\t" + response.getSignature());
```

示例表单HTML代码如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>

<form action="http://bucketname.your-endpoint/" method="post" enctype="multipart/form-data">
Object key
<!-- 对象名 -->
<input type="text" name="key" value="objectname" />
<p>
ACL
<!-- 对象ACL权限 -->
<input type="text" name="x-obs-acl" value="public-read" />
<p>
Content-Type
<!-- 对象MIME类型 -->
<input type="text" name="content-type" value="text/plain" />
<p>
<!-- policy的base64编码值 -->
<input type="hidden" name="policy" value="*** Provide your policy ***" />
<!-- AK -->
<input type="hidden" name="AccessKeyId" value="*** Provide your access key ***"/>
<!-- 签名串信息 -->
<input type="hidden" name="signature" value="*** Provide your signature ***"/>

<input name="file" type="file" />
<input name="submit" value="Upload" type="submit" />
```



```
</form>
</body>
</html>
```

说明

- HTML表单中的policy，signature的值均是从ObsClient.createPostSignature的返回结果中获取。
- 您可以直接下载表单HTML示例**PostDemo**。

7 下载对象

7.1 对象下载简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS Android SDK提供了丰富的对象下载接口，可以通过以下方式下载对象：

- **流式下载**
- **范围下载**
- **断点续传下载**

您可以通过ObsClient.getObject下载对象。

7.2 流式下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

以下代码展示了如何进行流式下载：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```

```
ObsObject obsObject = obsClient.getObject("bucketname", "objectname");

// 读取对象内容
Log.i("GetObject", "Object content:");
InputStream input = obsObject.getObjectContent();
byte[] b = new byte[1024];
ByteArrayOutputStream bos = new ByteArrayOutputStream();
int len;
while ((len=input.read(b)) != -1){
    bos.write(b, 0, len);
}

Log.i("GetObject", new String(bos.toByteArray()));
bos.close();
input.close();
```

说明

- ObsClient.getObject的返回实例ObsObject实例包含对象所在的桶、对象名、对象属性、对象输入流等。
- 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。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息，。如您的元数据存储的 content-disposition 为 “attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。
- 如果不需要 SDK 帮您解码，可以调用 GetObjectRequest.setIsEncodeHeaders(false) 关闭自动解码。
- 您也可以通过 obsObject.getMetadata().getOriginalHeaders() 获取所有原始响应头的信息。

须知

ObsObject.getObjectContent获取的对象输入流一定要显式关闭，否则会造成资源泄露。

7.3 范围下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

如果只需要下载对象的其中一部分数据，可以使用范围下载，下载指定范围的数据。如果指定的下载范围是0~1000，则返回第0到第1000个字节的数据，包括第1000个，共1001字节的数据，即[0, 1000]。如果指定的范围无效，则返回整个对象的数据。以下代码展示了如何进行范围下载：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
```



```
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
// 指定开始和结束范围
request.setRangeStart(0l);
request.setRangeEnd(1000l);
ObsObject obsObject = obsClient.getObject(request);

// 读取数据
byte[] buf = new byte[1024];
InputStream in = obsObject.getObjectContent();
for (int n = 0; n != -1; ) {
    n = in.read(buf, 0, buf.length);
}

in.close();
```

说明

- 如果指定的范围无效（比如开始位置、结束位置为负数，大于文件大小），则会返回整个对象。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息，。如您的元数据存储的 content-disposition 为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。
- 如果不需要 SDK 帮您解码，可以调用 GetObjectRequest.setIsEncodeHeaders(false) 关闭自动解码。
- 您也可以通过 obsObject.getMetadata().getOriginalHeaders() 获取所有原始响应头的信息。
- 可以利用范围下载并发下载大对象，详细代码示例请参考 **ConcurrentDownloadObjectSample**。

7.4 获取下载进度

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过GetObjectRequest.setProgressInterval设置数据传输接口来获取下载的进度。

本示例用于下载桶名为“bucketname”里，名称为“objectname”的对象并通过ProgressInterval监控下载进度。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
```

```
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
request.setProgressListener(new ProgressListener() {

    @Override
    public void progressChanged(ProgressStatus status) {
        // 获取下载平均速率
        Log.i("GetObject", "AverageSpeed:" + status.getAverageSpeed());
        // 获取下载进度百分比
        Log.i("GetObject", "TransferPercentage:" + status.getTransferPercentage());
    }
});
// 每上传1MB数据反馈下载进度
request.setProgressInterval(1024 * 1024L);
ObsObject obsObject = obsClient.getObject(request);

// 读取对象内容
System.out.println("Object content:");
InputStream input = obsObject.getObjectContent();
byte[] b = new byte[1024];
ByteArrayOutputStream bos = new ByteArrayOutputStream();
int len;
while ((len=input.read(b)) != -1){
    bos.write(b, 0, len);
}

System.out.println(new String(bos.toByteArray()));
bos.close();
input.close();
```

说明

- 支持获取下载进度的接口包括：流式下载、范围下载和断点续传下载。

7.5 限定条件下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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，否则抛出异常，下载对象失败。

您可以使用的限定条件如下：

参数	作用	OBS Android SDK对应方法
If-Modified-Since	如果对象在指定的时间后有修改，则返回对象内容，否则返回错误。	GetObjectRequest.setIfModifiedSince
If-Unmodified-Since	如果对象在指定的时间后没有修改，则返回对象内容，否则返回错误。	GetObjectRequest.setIfUnmodifiedSince

参数	作用	OBS Android SDK对应方法
If-Match	如果对象的ETag值与该参数值相同，则返回对象内容，否则抛出异常。	GetObjectRequest.setIfMatchTag
If-None-Match	如果对象的ETag值与该参数值不相同，则返回对象内容，否则抛出异常。	GetObjectRequest.setIfNoneMatchTag

说明

- 对象的ETag值是指对象数据的MD5校验值。
- 如果包含If-Unmodified-Since并且不符合或者包含If-Match并且不符合，抛出异常中HTTP状态码为：412 precondition failed。
- 如果包含If-Modified-Since并且不符合或者包含If-None-Match并且不符合，抛出异常中HTTP状态码为：304 Not Modified。

以下代码展示了如何进行限定条件下载：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
request.setRangeStart(0l);
request.setRangeEnd(1000l);

request.setIfModifiedSince(new SimpleDateFormat("yyyy-MM-dd").parse("2016-01-01"));
ObsObject obsObject = obsClient.getObject(request);

obsObject.getObjectContent().close();
```

7.6 重写响应头

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

下载对象时，可以重写部分HTTP/HTTPS响应头信息。可重写的响应头信息见下表：

参数	作用	OBS Android SDK对应方法
contentType	重写HTTP/HTTPS响应中的Content-Type	ObjectRepleaceMetadata.setContentType
contentLanguage	重写HTTP/HTTPS响应中的Content-Language	ObjectRepleaceMetadata.setContentLanguage
expires	重写HTTP/HTTPS响应中的Expires	ObjectRepleaceMetadata.setExpires
cacheControl	重写HTTP/HTTPS响应中的Cache-Control	ObjectRepleaceMetadata.setCacheControl
contentDisposition	重写HTTP/HTTPS响应中的Content-Disposition	ObjectRepleaceMetadata.setContentDisposition
contentEncoding	重写HTTP/HTTPS响应中的Content-Encoding	ObjectRepleaceMetadata.setContentEncoding

以下代码展示了如何重写响应头：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
ObjectRepleaceMetadata replaceMetadata = new ObjectRepleaceMetadata();
replaceMetadata.setContentType("image/jpeg");
request.setReplaceMetadata(replaceMetadata);

ObsObject obsObject = obsClient.getObject(request);
Log.i("GetObject", object.getMetadata().getContentType());

obsObject.getObjectContent().close();
```

7.7 获取自定义元数据

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

下载对象成功后会返回对象的自定义元数据。以下代码展示了如何获取自定义元数据：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 上传对象，设置自定义元数据
PutObjectRequest request = new PutObjectRequest("bucketname", "objectname");
ObjectMetadata metadata = new ObjectMetadata();
metadata.addUserMetadata("property", "property-value");
request.setMetadata(metadata);
obsClient.putObject(request);
// 下载对象，获取对象自定义元数据
GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
ObsObject obsObject = obsClient.getObject(request);
// 读取对象元数据
System.out.println(obsObject.getMetadata().getContentType());
System.out.println(obsObject.getMetadata().getUserMetadata("property"));

obsObject.getObjectContent().close();
```

说明

- ObsClient.getObject的返回实例ObsObject实例包含对象所在的桶、对象名、对象属性、对象输入流等。
- 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。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息，。如您的元数据存储的 content-disposition 为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。
- 如果不需要 SDK 帮您解码，可以调用 GetObjectRequest.setIsEncodeHeaders(false) 关闭自动解码。
- 您也可以通过 obsObject.getMetadata().getOriginalHeaders() 获取所有原始响应头的信息。

7.8 下载归档存储对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

如果要下载归档存储对象，需要先将归档存储对象恢复。恢复归档存储对象的恢复选项可支持两类，见下表：

选项	说明	OBS Android SDK对应值
快速恢复	恢复耗时1~5分钟。	RestoreTierEnum.EXPEDITED
标准恢复	恢复耗时3~5小时。默认值。	RestoreTierEnum.STANDARD

⚠ 注意

重复恢复归档存储数据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。产生的标准存储类型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。

您可以通过ObsClient.restoreObject恢复归档存储对象。以下代码展示了如何下载归档存储对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

RestoreObjectRequest request = new RestoreObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("objectname");
request.setDays(1);
request.setRestoreTier(RestoreTierEnum.EXPEDITED);
obsClient.restoreObject(request);

// 等待对象恢复
Thread.sleep(60 * 6 * 1000);

// 下载对象
ObsObject obsObject = obsClient.getObject("bucketname", "objectname");

obsObject.getObjectContent().close();
```

说明

- ObsClient.restoreObject中指定的对象必须是归档存储类型，否则调用该接口会抛出异常。
- RestoreObjectRequest.setDays指定恢复对象保存的时间，取值范围是1~30。
- RestoreObjectRequest.setTier指定恢复选项，表示恢复对象所耗的时间。

7.9 断点续传下载

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

当下载大对象到本地时，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，否则抛出异常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ObsClient.downloadFile进行断点续传下载。该接口可设置的参数如下：

参数	作用	OBS Android SDK对应方法
bucketName	桶名，必选参数。	DownloadFileRequest.setBucketName
objectKey	对象名，必选参数。	DownloadFileRequest.setObjectKey
downloadFile	下载对象的本地文件全路径。当该值为空时，默认为当前程序的运行目录。	DownloadFileRequest.setDownloadFile
partSize	分段大小，单位字节，取值范围是100KB~5GB，默认为5MB。	DownloadFileRequest.setPartSize
taskNum	分段下载时的最大并发数，默认为1。	DownloadFileRequest.setTaskNum
isEncodeHeaders	是否自动解码响应头	DownloadFileRequest.setIsEncodeHeaders
enableCheckpoint	是否开启断点续传模式，默认为false，表示不开启。	DownloadFileRequest.setEnableCheckpoint
checkpointFile	记录下载进度的文件，只在断点续传模式下有效。当该值为空时，默认与下载对象的本地文件路径同目录。	DownloadFileRequest.setCheckpointFile
versionId	对象的版本号。	DownloadFileRequest.setVersionId
ifModifiedSince	如果对象在指定的时间后有修改，则返回对象内容，否则返回错误。	DownloadFileRequest.setIfModifiedSince
ifUnmodifiedSince	如果对象在指定的时间后没有修改，则返回对象内容，否则返回错误。	DownloadFileRequest.setIfUnmodifiedSince
ifMatchTag	如果对象的ETag值与该参数值相同，则返回对象内容，否则抛出异常。	DownloadFileRequest.setIfMatchTag
ifNoneMatchTag	如果对象的ETag值与该参数值不相同，则返回对象内容，否则抛出异常。	DownloadFileRequest.setIfNoneMatchTag
progressListener	设置数据传输监听器，用于获取下载进度。	DownloadFileRequest.setProgressListener

以下代码展示了如何使用断点续传下载接口下载对象到本地文件：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
DownloadFileRequest request = new DownloadFileRequest("bucketname", "objectname");
// 设置下载对象的本地文件路径
request.setDownloadFile("localfile");
// 设置分段下载时的最大并发数
request.setTaskNum(5);
// 设置分段大小为10MB
request.setPartSize(10 * 1024 * 1024);
// 开启断点续传模式
request.setEnableCheckpoint(true);
try{
    // 进行断点续传下载
    DownloadFileResult result = obsClient.downloadFile(request);
}catch (ObsException e) {
    // 发生异常时可再次调用断点续传下载接口进行重新下载
}
```

说明

- 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重下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。
- **enableCheckpoint参数**默认是false，代表不启用断点续传模式，此时断点续传下载接口退化成对范围下载的简单封装，不会产生checkpoint文件。
- **checkpointFile参数**仅在**enableCheckpoint参数**为true时有效。
- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息，。如您的元数据存储的 content-disposition 为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。
- 如果不需要 SDK 帮您解码，可以调用 DownloadFileRequest.setIsEncodeHeaders(false) 关闭自动解码。

7.10 图片处理

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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。

更多关于图片处理的内容，参见**图片处理特性指南**。

以下代码展示了如何使用下载对象接口实现图片处理：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);
GetObjectRequest request = new GetObjectRequest("bucketname", "objectname.jpg");
// 对图片依次进行缩放、旋转
request.setImageProcess("image/resize,m_fixed,w_100,h_100/rotate,90");
ObsObject obsObject = obsClient.getObject(request);

obsObject.getObjectContent().close();
```

说明

- 使用GetObjectRequest.setImageProcess指定图片处理参数。
- 图片处理参数支持级联处理，可对图片文件依次执行多条命令。

8 管理对象

8.1 设置对象属性

您可以通过ObsClient.setObjectMetadata来设置对象属性，包括对象自定义元数据等信息。

除了 HTTP 标准头域外，您也可以指定自定义元数据，自定义元数据仅支持英文字符、数字与中划线【 - 】。

以下代码展示了如何设置对象属性：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

SetObjectMetadataRequest request = new SetObjectMetadataRequest("bucketname", "objectname");
// 对象的 HTTP 标准头域
request.setContentType("ContentType");
request.setExpires("Expires");
// 设置自定义元数据
request.addUserMetadata("property1", "property-value1");

ObjectMetadata metadata = obsClient.setObjectMetadata(request);

Log.i("\t" + metadata.getUserMetadata("property1"));
```

说明

- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在发送请求时对您头域中的**中文汉字**进行 url 编码，发送编码后数据。如您设置的值 content-disposition 为“attachment; filename="中文.txt"”，则对象元数据中存储的信息为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”。使用浏览器访问时浏览器将会自动解码。
- 如果不需要 SDK 帮您编码，可以调用 SetObjectMetadataRequest.setIsEncodeHeaders(false) 关闭自动编码。

8.2 获取对象属性

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getObjectMetadata来获取对象属性，包括对象长度，对象MIME类型，对象自定义元数据等信息。以下代码展示了如何获取对象属性：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObjectMetadata metadata = obsClient.getObjectMetadata("bucketname", "objectname");
Log.i("GetObjectMetadata","\t" + metadata.getContentType());
Log.i("GetObjectMetadata","\t" + metadata.getContentLength());
// 获取用户自定义元数据
Log.i("GetObjectMetadata","\t" + metadata.getUserMetadata("property"));
// 获取所有原始响应头域
Log.i("GetObjectMetadata","\t" + metadata.getOriginalHeaders());
```

说明

- 由于 HTTP 编码规范限制，无法发送非 ASCII 码字符，SDK 会在接收响应时使用 url 解码规则解码响应头中的信息。如您的元数据存储的 content-disposition 为“attachment; filename="%E4%B8%AD%E6%96%87.txt"”，则 SDK 获取结果为“attachment; filename="中文.txt"”。
- 如果不需要 SDK 帮您解码，可以调用 GetObjectMetadataRequest.setIsEncodeHeaders(false) 关闭自动解码。
- 您也可以通过 metadata.getOriginalHeaders 获取所有原始响应头的信息。

8.3 管理对象访问权限

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

对象访问权限与桶访问权限类似，也可支持预定义访问策略（参见**桶访问权限**）或直接设置。

对象访问权限（**ACL**）可以通过三种方式设置：

1. 上传对象时指定预定义访问策略。

2. 调用ObsClient.setObjectAcl指定预定义访问策略。
3. 调用ObsClient.setObjectAcl直接设置。

上传对象时指定预定义访问策略

以下代码展示如何在上传对象时指定预定义访问策略：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

PutObjectRequest request = new PutObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("objectname");
request.setFile(new File("localfile")); // localfile为待上传的本地文件路径，需要指定到具体的文件名
// 设置对象访问权限为公共读
request.setAcl(AccessControlList.REST_CANNED_PUBLIC_READ);
obsClient.putObject(request);
```

为对象设置预定义访问策略

以下代码展示如何为对象设置预定义访问策略：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 设置对象访问权限为私有读写
obsClient.setObjectAcl("bucketname", "objectname", AccessControlList.REST_CANNED_PRIVATE);
```

直接设置对象访问权限

以下代码展示如何直接设置对象访问权限：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
owner.setId("ownerid");
acl.setOwner(owner);
```



```
// 为指定用户设置完全控制权限
acl.grantPermission(new CanonicalGrantee("userid"), Permission.PERMISSION_FULL_CONTROL);
// 为所有用户设置读权限
acl.grantPermission(GroupGrantee.ALL_USERS, Permission.PERMISSION_READ);
obsClient.setObjectAcl("bucketname", "objectname", acl);
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取对象访问权限

您可以通过ObsClient.getObjectAcl获取对象的访问权限。以下代码展示如何获取对象访问权限：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AccessControlList acl = obsClient.getObjectAcl("bucketname", "objectname");
Log.i("GetObjectAcl", acl);
```

8.4 列举对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.listObjects列举出桶里的对象。

该接口可设置的参数如下：

参数	作用	OBS Android SDK对应方法
bucketName	桶名。	ListObjectsRequest.setBucketName
prefix	限定返回的对象名必须带有prefix前缀。	ListObjectsRequest.setPrefix
marker	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。	ListObjectsRequest.setMarker

参数	作用	OBS Android SDK对应方法
maxKeys	列举对象的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。	ListObjectsRequest.setMaxKeys
delimiter	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。	ListObjectsRequest.setDelimiter

简单列举

以下代码展示如何简单列举对象，最多返回1000个对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ObjectListing result = obsClient.listObjects("bucketname");
for(ObsObject obsObject : result.getObjects()){
    Log.i("ListObjects", "\t" + obsObject.getObjectKey());
    Log.i("ListObjects","\t" + obsObject.getOwner());
}
```

说明

- 每次至多返回1000个对象，如果指定桶包含的对象数量大于1000，则返回结果中ObjectListing.isTruncated为true表明本次没有返回全部对象，并可通过ObjectListing.getNextMarker获取下次列举的起始位置。
- 如果想获取指定桶包含的所有对象，可以采用分页列举的方式。

指定数目列举

以下代码展示如何指定数目列举对象：

```
// 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_ID。
```



```
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
// 只列举100个对象
request.setMaxKeys(100);
ObjectListing result = obsClient.listObjects(request);
for(ObsObject obsObject : result.getObjects()){
    Log.i("ListObjects","\t" + obsObject.getObjectKey());
    Log.i("ListObjects","\t" + obsObject.getOwner());
}
```

指定前缀列举

以下代码展示如何指定前缀列举对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
// 设置列举带有prefix前缀的100个对象
request.setMaxKeys(100);
request.setPrefix("prefix");
ObjectListing result = obsClient.listObjects(request);
for(ObsObject obsObject : result.getObjects()){
    Log.i("ListObjects","\t" + obsObject.getObjectKey());
    Log.i("ListObjects","\t" + obsObject.getOwner());
}
```

指定起始位置列举

以下代码展示如何指定起始位置列举对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
// 设置列举对象名字典序在"test"之后的100个对象
request.setMaxKeys(100);
request.setMarker("test");
ObjectListing result = obsClient.listObjects(request);
for(ObsObject obsObject : result.getObjects()){
    Log.i("ListObjects","\t" + obsObject.getObjectKey());
    Log.i("ListObjects","\t" + obsObject.getOwner());
}
```


分页列举全部对象

以下代码展示分页列举全部对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
// 设置每页100个对象
request.setMaxKeys(100);

ObjectListing result;
do{
    result = obsClient.listObjects(request);
    for(ObsObject obsObject : result.getObjects()){
        Log.i("ListObjects","\t" + obsObject.getObjectKey());
        Log.i("ListObjects","\t" + obsObject.getOwner());
    }

    request.setMarker(result.getNextMarker());
}while(result.isTruncated());
```

列举文件夹中的所有对象

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。文件夹对象实际上是一个大小为0且对象名以“/”结尾的对象，将这个文件夹对象名作为前缀，即可模拟列举文件夹中对象的功能。以下代码展示如何列举文件夹中的对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
// 设置文件夹对象名"dir/"为前缀
request.setPrefix("dir/");
request.setMaxKeys(1000);

ObjectListing result;

do{
    result = obsClient.listObjects(request);
    for (ObsObject obsObject : result.getObjects())
    {
        Log.i("ListObjects","\t" + obsObject.getObjectKey());
        Log.i("ListObjects","\t" + obsObject.getOwner());
    }
    request.setMarker(result.getNextMarker());
}while(result.isTruncated());
```

按文件夹分组列举所有对象

以下代码展示如何按文件夹分组，列举桶内所有对象：


```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListObjectsRequest request = new ListObjectsRequest("bucketname");
request.setMaxKeys(1000);
// 设置文件夹分隔符"/"
request.setDelimiter("/");
ObjectListing result = obsClient.listObjects(request);
Log.i("ListObjects", "Objects in the root directory:");
for(ObsObject obsObject : result.getObjects()){
    Log.i("ListObjects","\t" + obsObject.getObjectKey());
    Log.i("ListObjects","\t" + obsObject.getOwner());
}
listObjectsByPrefix(obsClient, request, result);
```

递归列出子文件夹中对象的函数*listObjectsByPrefix*的示例代码如下：

```
static void listObjectsByPrefix(ObsClient obsClient, ListObjectsRequest request, ObjectListing result) throws ObsException
{
    for(String prefix : result.getCommonPrefixes()){
        Log.i("ListObjects", "Objects in folder [" + prefix + "]:");
        request.setPrefix(prefix);
        result  = obsClient.listObjects(request);
        for(ObsObject obsObject : result.getObjects()){
            Log.i("ListObjects","\t" + obsObject.getObjectKey());
            Log.i("ListObjects","\t" + obsObject.getOwner());
        }
        listObjectsByPrefix(obsClient, request, result);
    }
}
```

说明

- 以上代码示例没有考虑文件夹中对象数超过1000个的情况。
- 由于是需要列举出文件夹中的对象和子文件夹，且文件夹对象总是以“/”结尾，因此delimiter总是为“/”。
- 每次递归的返回结果中ObjectListing.getObjects包含的是文件夹中的对象；ObjectListing.getCommonPrefixes包含的是文件夹的子文件夹。

8.5 删除对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

说明

请您谨慎使用删除操作，如果对象所在的桶未开启多版本控制功能，该对象一旦删除将无法恢复。

删除单个对象

您可以通过ObsClient.deleteObject删除单个对象。以下代码展示如何删除单个对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
obsClient.deleteObject("bucketname", "objectname");
```

批量删除对象

您可以通过ObsClient.deleteObjects批量删除对象。

每次最多删除1000个对象，并支持两种响应模式：详细（verbose）模式和简单（quiet）模式。

- 详细模式：返回的删除成功和删除失败的所有结果，默认模式。
- 简单模式：只返回的删除过程中出错的结果。

以下代码展示了如何进行批量删除桶内所有对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListVersionsRequest request = new ListVersionsRequest("bucketname");
// 每次批量删除100个对象
request.setMaxKeys(100);
ListVersionsResult result;
do {
    result = obsClient.listVersions(request);

    DeleteObjectsRequest deleteRequest = new DeleteObjectsRequest("bucketname");

    for(VersionOrDeleteMarker v  : result.getVersions()) {
        deleteRequest.addKeyAndVersion(v.getKey(), v.getVersionId());
    }

    DeleteObjectsResult deleteResult = obsClient.deleteObjects(deleteRequest);
    // 获取删除成功的对象
    Log.i("DeletesObjects",deleteResult.getDeletedObjectResults());
    // 获取删除失败的对象
    Log.i("DeletesObjects", deleteResult.getErrorResults());

    request.setKeyMarker(deleteResult.getNextKeyMarker());
    request.setVersionIdMarker(deleteResult.getNextVersionIdMarker());
}while(result.isTruncated());
```


8.6 复制对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

复制对象特性用来为OBS上已经存在的对象创建一个副本。

您可以通过ObsClient.copyObject来复制对象。复制对象时，可重新指定新对象的属性和设置对象权限，且支持条件复制。

说明

- 如果待复制的源对象是归档存储类型，则必须先恢复源对象才能进行复制。

简单复制

以下代码展示了如何进行简单复制：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

CopyObjectResult result = obsClient.copyObject("sourcebucketname", "sourceobjectname",
"destbucketname", "destobjectname");
Log.i("CopyObject","\t" + result.getEtag());
```

重写对象属性

以下代码展示了如何在复制对象时重写对象属性：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

CopyObjectRequest request = new CopyObjectRequest("sourcebucketname", "sourceobjectname",
"destbucketname", "destobjectname");
// 设置进行对象属性重写
request.setReplaceMetadata(true);
ObjectMetadata newObjectMetadata = new ObjectMetadata();
newObjectMetadata.setContentType("image/jpeg");
newObjectMetadata.addUserMetadata("property", "property-value");
newObjectMetadata.setObjectStorageClass(StorageClassEnum.WARM);
request.setNewObjectMetadata(newObjectMetadata);
```

```
CopyObjectResult result = obsClient.copyObject(request);
Log.i("CopyObject","\t" + result.getEtag());
```

说明

CopyObjectRequest.setReplaceMetadata需与CopyObjectRequest.setNewObjectMetadata配合使用。

限定条件复制

复制对象时，可以指定一个或多个限定条件，满足限定条件时则进行复制，否则抛出异常，复制对象失败。

您可以使用的限定条件如下：

参数	作用	OBS Android SDK对应方法
Copy-Source-If-Modified-Since	如果源对象在指定的时间后有修改，则进行复制，否则抛出异常。	CopyObjectRequest.setIfModifiedSince
Copy-Source-If-Unmodified-Since	如果源对象在指定的时间后没有修改，则进行复制，否则抛出异常。	CopyObjectRequest.setIfUnmodifiedSince
Copy-Source-If-Match	如果源对象的ETag值与该参数值相同，则进行复制，否则抛出异常。	CopyObjectRequest.setIfMatchTag
Copy-Source-If-None-Match	如果源对象的ETag值与该参数值不相同，则进行复制，否则抛出异常。	CopyObjectRequest.setIfNoneMatchTag

说明

- 源对象的ETag值是指源对象数据的MD5校验值。
- 如果包含Copy-Source-If-Unmodified-Since并且不符合，或者包含Copy-Source-If-Match并且不符合，或者包含Copy-Source-If-Modified-Since并且不符合，或者包含Copy-Source-If-None-Match并且不符合，则复制失败，抛出异常中HTTP状态码为：412 precondition failed。
- Copy-Source-If-Modified-Since和Copy-Source-If-None-Match可以一起使用；Copy-Source-If-Unmodified-Since和Copy-Source-If-Match可以一起使用。

以下代码展示了如何进行限定条件复制：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```

```
CopyObjectRequest request = new CopyObjectRequest("sourcebucketname", "sourceobjectname",
"destbucketname", "destobjectname");

request.setIfModifiedSince(new SimpleDateFormat("yyyy-MM-dd").parse("2016-01-01"));
request.setIfNoneMatchTag("none-match-etag");

CopyObjectResult result = obsClient.copyObject(request);
Log.i("CopyObject","\t" + result.getEtag());
```

重写对象访问权限

以下代码展示了如何在复制对象时重写对象访问权限：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

CopyObjectRequest request = new CopyObjectRequest("sourcebucketname", "sourceobjectname",
"destbucketname", "destobjectname");

// 复制时重写对象访问权限为公共读
request.setAcl(AccessControlList.REST_CANNED_PUBLIC_READ);
CopyObjectResult result = obsClient.copyObject(request);
Log.i("CopyObject","\t" + result.getEtag());
```


9 临时授权访问

9.1 使用临时 URL 进行授权访问

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS客户端支持通过访问密钥、请求方法类型、请求参数等信息生成一个在Query参数中携带鉴权信息的URL，可将该URL提供给其他用户进行临时访问。在生成URL时，您需要指定URL的有效期来限制访客用户的访问时长。

如果您想授予其他用户对桶或对象临时进行其他操作的权限（例如上传或下载对象），则需要生成带对应请求的URL后（例如使用生成PUT请求的URL上传对象），将该URL提供给其他用户。

通过该方式可支持的操作以及相关信息见下表：

操作名	HTTP请求方法（OBS Android SDK对应值）	特殊操作符（OBS Android SDK对应值）	是否需要桶名	是否需要对象名
创建桶	HttpMethodEnum.PUT	N/A	是	否
获取桶列表	HttpMethodEnum.GET	N/A	否	否
删除桶	HttpMethodEnum.DELETE	N/A	是	否
列举桶内对象	HttpMethodEnum.GET	N/A	是	否
列举桶内多版本对象	HttpMethodEnum.GET	SpecialParamEnum.VERSIONS	是	否

操作名	HTTP请求方法（OBS Android SDK对应值）	特殊操作符（OBS Android SDK对应值）	是否需要桶名	是否需要对象名
列举分段上传任务	HttpMethodEnum.GET	SpecialParamEnum.UPLOADS	是	否
获取桶元数据	HttpMethodEnum.HEAD	N/A	是	否
获取桶区域位置	HttpMethodEnum.GET	SpecialParamEnum.LOCATION	是	否
获取桶存量信息	HttpMethodEnum.GET	SpecialParamEnum.STORAGEINFO	是	否
设置桶配额	HttpMethodEnum.PUT	SpecialParamEnum.QUOTA	是	否
获取桶配额	HttpMethodEnum.GET	SpecialParamEnum.QUOTA	是	否
设置桶存储类型	HttpMethodEnum.PUT	SpecialParamEnum.STORAGEPOLICY	是	否
获取桶存储类型	HttpMethodEnum.GET	SpecialParamEnum.STORAGEPOLICY	是	否
设置桶访问权限	HttpMethodEnum.PUT	SpecialParamEnum.ACL	是	否
获取桶访问权限	HttpMethodEnum.GET	SpecialParamEnum.ACL	是	否
开启/关闭桶日志	HttpMethodEnum.PUT	SpecialParamEnum.LOGGING	是	否
查看桶日志	HttpMethodEnum.GET	SpecialParamEnum.LOGGING	是	否
设置桶策略	HttpMethodEnum.PUT	SpecialParamEnum.POLICY	是	否
查看桶策略	HttpMethodEnum.GET	SpecialParamEnum.POLICY	是	否
删除桶策略	HttpMethodEnum.DELETE	SpecialParamEnum.POLICY	是	否
设置生命周期规则	HttpMethodEnum.PUT	SpecialParamEnum.LIFECYCLE	是	否
查看生命周期规则	HttpMethodEnum.GET	SpecialParamEnum.LIFECYCLE	是	否
删除生命周期规则	HttpMethodEnum.DELETE	SpecialParamEnum.LIFECYCLE	是	否

操作名	HTTP请求方法（OBS Android SDK对应值）	特殊操作符（OBS Android SDK对应值）	是否需要桶名	是否需要对象名
设置托管配置	HttpMethodEnum.PUT	SpecialParamEnum.WEBSITE	是	否
查看托管配置	HttpMethodEnum.GET	SpecialParamEnum.WEBSITE	是	否
清除托管配置	HttpMethodEnum.DELETE	SpecialParamEnum.WEBSITE	是	否
设置桶多版本状态	HttpMethodEnum.PUT	SpecialParamEnum.VERSIONING	是	否
查看桶多版本状态	HttpMethodEnum.GET	SpecialParamEnum.VERSIONING	是	否
设置跨域规则	HttpMethodEnum.PUT	SpecialParamEnum.CORS	是	否
查看跨域规则	HttpMethodEnum.GET	SpecialParamEnum.CORS	是	否
删除跨域规则	HttpMethodEnum.DELETE	SpecialParamEnum.CORS	是	否
设置桶标签	HttpMethodEnum.PUT	SpecialParamEnum.TAGGING	是	否
查看桶标签	HttpMethodEnum.GET	SpecialParamEnum.TAGGING	是	否
删除桶标签	HttpMethodEnum.DELETE	SpecialParamEnum.TAGGING	是	否
上传对象	HttpMethodEnum.PUT	N/A	是	是
追加上传	HttpMethodEnum.POST	SpecialParamEnum.APPEND	是	是
下载对象	HttpMethodEnum.GET	N/A	是	是
复制对象	HttpMethodEnum.PUT	N/A	是	是
删除对象	HttpMethodEnum.DELETE	N/A	是	是
批量删除对象	HttpMethodEnum.POST	SpecialParamEnum.DELETE	是	是
获取对象属性	HttpMethodEnum.HEAD	N/A	是	是

操作名	HTTP请求方法（OBS Android SDK对应值）	特殊操作符（OBS Android SDK对应值）	是否需要桶名	是否需要对象名
设置对象访问权限	HttpMethodEnum.PUT	SpecialParamEnum.ACL	是	是
查看对象访问权限	HttpMethodEnum.GET	SpecialParamEnum.ACL	是	是
初始化分段上传任务	HttpMethodEnum.POST	SpecialParamEnum.UPLOADS	是	是
上传段	HttpMethodEnum.PUT	N/A	是	是
复制段	HttpMethodEnum.PUT	N/A	是	是
列举已上传的段	HttpMethodEnum.GET	N/A	是	是
合并段	HttpMethodEnum.POST	N/A	是	是
取消分段上传任务	HttpMethodEnum.DELETE	N/A	是	是
恢复归档存储对象	HttpMethodEnum.POST	SpecialParamEnum.RESTORE	是	是

通过OBS Android SDK生成临时URL访问OBS的步骤如下：

步骤1 通过ObsClient.createTemporarySignature生成带签名信息的URL。

步骤2 使用任意HTTP库发送HTTP/HTTPS请求，访问OBS服务。

----结束

⚠ 注意

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：

1. 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**。
2. 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。

以下代码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。

创建桶

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// URL有效期，3600秒
long expireSeconds = 3600L;

TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.PUT, expireSeconds);
request.setBucketName("bucketname");
TemporarySignatureResponse response = obsClient.createTemporarySignature(request);
Log.i("CreateTemporarySignature", "Creating bucket using temporary signature url:");
Log.i("CreateTemporarySignature", "\t" + response.getSignedUrl());

Request.Builder builder = new Request.Builder();
for (Map.Entry<String, String> entry : response.getActualSignedRequestHeaders().entrySet()) {
    builder.header(entry.getKey(), entry.getValue());
}
// 使用PUT请求创建桶
String location = "your bucket location";
Request httpRequest = builder.url(response.getSignedUrl()).put(RequestBody.create(null, "<CreateBucketConfiguration><LocationConstraint>" + location + "</LocationConstraint></CreateBucketConfiguration>".getBytes())).build();
OkHttpClient httpClient = new OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
            .cache(null).build();

Call c = httpClient.newCall(httpRequest);
Response res = c.execute();
Log.i("CreateTemporarySignature", "\tStatus:" + res.code());
if (res.body() != null) {
    Log.i("CreateTemporarySignature", "\tContent:" + res.body().string() + "\n");
}
res.close();
```

上传对象

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// URL有效期，3600秒
long expireSeconds = 3600L;

Map<String, String> headers = new HashMap<String, String>();
String contentType = "text/plain";
headers.put("Content-Type", contentType);

TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.PUT, expireSeconds);
request.setBucketName("bucketname");
request.setObjectKey("objectname");
request.setHeaders(headers);

TemporarySignatureResponse response = obsClient.createTemporarySignature(request);
```



```
Log.i("CreateTemporarySignature", "Creating object using temporary signature url:");
Log.i("CreateTemporarySignature", "\t" + response.getSignedUrl());
Request.Builder builder = new Request.Builder();
for (Map.Entry<String, String> entry : response.getActualSignedRequestHeaders().entrySet()) {
    builder.header(entry.getKey(), entry.getValue());
}

//使用PUT请求上传对象
Request httpRequest =
builder.url(response.getSignedUrl()).put(RequestBody.create(MediaType.parse(contentType), "Hello
OBS".getBytes("UTF-8"))).build();
OkHttpClient httpClient = new
OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
        .cache(null).build();

Call c = httpClient.newCall(httpRequest);
Response res = c.execute();
Log.i("CreateTemporarySignature", "\tStatus:" + res.code());
if (res.body() != null) {
    Log.i("CreateTemporarySignature", "\tContent:" + res.body().string() + "\n");
}
res.close();
```

下载对象

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// URL有效期，3600秒
long expireSeconds = 3600L;


TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.GET,
expireSeconds);
request.setBucketName("bucketname");
request.setObjectKey("objectname");

TemporarySignatureResponse response = obsClient.createTemporarySignature(request);

Log.i("CreateTemporarySignature", "Getting object using temporary signature url:");
Log.i("CreateTemporarySignature", "\t" + response.getSignedUrl());
Request.Builder builder = new Request.Builder();
for (Map.Entry<String, String> entry : response.getActualSignedRequestHeaders().entrySet()) {
    builder.header(entry.getKey(), entry.getValue());
}

//使用GET请求下载对象
Request httpRequest = builder.url(response.getSignedUrl()).get().build();
OkHttpClient httpClient = new
OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
        .cache(null).build();

Call c = httpClient.newCall(httpRequest);
Response res = c.execute();
Log.i("CreateTemporarySignature", "\tStatus:" + res.code());
if (res.body() != null) {
    Log.i("CreateTemporarySignature", "\tContent:" + res.body().string() + "\n");
}
res.close();
```


列举对象

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// URL有效期，3600秒
long expireSeconds = 3600L;


TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.GET, expireSeconds);
request.setBucketName("bucketname");

TemporarySignatureResponse response = obsClient.createTemporarySignature(request);

Log.i("CreateTemporarySignature", "Getting object list using temporary signature url:");
Log.i("CreateTemporarySignature", "\t" + response.getSignedUrl());
Request.Builder builder = new Request.Builder();
for (Map.Entry<String, String> entry : response.getActualSignedRequestHeaders().entrySet()) {
    builder.header(entry.getKey(), entry.getValue());
}

//使用GET请求获取对象列表
Request httpRequest = builder.url(response.getSignedUrl()).get().build();
OkHttpClient httpClient = new
OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
        .cache(null).build();

Call c = httpClient.newCall(httpRequest);
Response res = c.execute();
Log.i("CreateTemporarySignature", "\tStatus:" + res.code());
if (res.body() != null) {
    Log.i("CreateTemporarySignature", "\tContent:" + res.body().string() + "\n");
}
res.close();
```

删除对象

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
// URL有效期，3600秒
long expireSeconds = 3600L;


TemporarySignatureRequest request = new TemporarySignatureRequest(HttpMethodEnum.DELETE, expireSeconds);
request.setBucketName("bucketname");
request.setObjectKey("objectname");

TemporarySignatureResponse response = obsClient.createTemporarySignature(request);

Log.i("CreateTemporarySignature", "Deleting object using temporary signature url:");
Log.i("CreateTemporarySignature", "\t" + response.getSignedUrl());
Request.Builder builder = new Request.Builder();
```



```
for (Map.Entry<String, String> entry : response.getActualSignedRequestHeaders().entrySet()) {
    builder.header(entry.getKey(), entry.getValue());
}

//使用DELETE删除对象
Request httpRequest = builder.url(response.getSignedUrl()).delete().build();
OkHttpClient httpClient = new
OkHttpClient.Builder().followRedirects(false).retryOnConnectionFailure(false)
        .cache(null).build();

Call c = httpClient.newCall(httpRequest);
Response res = c.execute();
Log.i("CreateTemporarySignature", "\tStatus:" + res.code());
if (res.body() != null) {
    Log.i("CreateTemporarySignature", "\tContent:" + res.body().string() + "\n");
}
res.close();
```

说明

HttpMethodEnum是OBS Android SDK定义的枚举类型，代表请求方法类型。

10 多版本控制

10.1 多版本控制简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS支持保存一个对象的多个版本，您可以利用多版本控制，在一个桶中保留多个版本的对象。

多版本功能可以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。

在默认情况下，OBS中新创建的桶不会开启多版本功能，向同一个桶上传同名的对象时，新上传的对象将覆盖原有的对象。

更多关于多版本控制的内容请参见**多版本控制**。

10.2 设置桶多版本状态

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketVersioning设置桶的多版本状态。OBS中的桶支持两种多版本状态：

多版本状态	说明	OBS Android SDK对应值
启用状态	1. 上传对象时，系统为每一个对象创建一个唯一版本号，上传同名的对象将不再覆盖旧的对象，而是创建新的不同版本号的同名对象。 2. 可以指定版本号下载对象，不指定版本号默认下载最新对象。 3. 删除对象时可以指定版本号删除，不带版本号删除对象仅产生一个带唯一版本号的删除标记，并不删除对象。 4. 列出桶内对象列表（ObsClient.listObjects）时默认列出最新对象列表，可以指定列出桶内所有版本对象列表（ObsClient.listVersions）。 5. 除了删除标记外，每个版本的对象存储均需计费。	VersioningStatusEnum.ENABLED
暂停状态	1. 旧的版本数据继续保留。 2. 上传对象时创建对象的版本号为null，上传同名的对象将覆盖原有同名的版本号为null的对象。 3. 可以指定版本号下载对象，不指定版本号默认下载最新对象。 4. 删除对象时可以指定版本号删除，不带版本号删除对象将产生一个版本号为null的删除标记，并删除版本号为null的对象。 5. 除了删除标记外，每个版本的对象存储均需计费。	VersioningStatusEnum.SUSPENDED

以下代码展示了如何设置桶的多版本状态：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 启用桶多版本状态
obsClient.setBucketVersioning("bucketname", new
BucketVersioningConfiguration(VersioningStatusEnum.ENABLED));

// 暂停桶多版本状态
obsClient.setBucketVersioning("bucketname", new
BucketVersioningConfiguration(VersioningStatusEnum.SUSPENDED));
```

10.3 查看桶多版本状态

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketVersioning查看桶的多版本状态。

本示例用于获取桶名为“bucketname”里的多版本状态。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketVersioningConfiguration status = obsClient.getBucketVersioning("bucketname");
Log.i("GetBucketVersioning", "\t" + status.getVersioningStatus());
```

说明

- 获取桶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服务端错误码**。

10.4 获取多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getObject接口传入版本号（versionId）来获取多版本对象。

本示例用于下载桶名为“bucketname”里，名称为“objectname”，指定VersionId的对象。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
```

```
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 设置versionId获取多版本对象
ObsObject obsObject = obsClient.getObject("bucketname", "objectname", "versionid");
obsObject.getObjectContent().close();
```

说明

- 如果版本号为空则默认下载最新版本的对象。

10.5 复制多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.copyObject接口传入版本号（versionId）来复制多版本对象。

本示例用于通过设置versionId复制sourcebucketname桶中sourceobjectname的多版本对象，复制到destbucketname桶中destobjectname。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

CopyObjectRequest request = new CopyObjectRequest();
request.setSourceBucketName("sourebucketname");
request.setSourceObjectKey("sourceobjectname");
// 设置要复制对象的版本号
request.setVersionId("versionid");
request.setDestinationBucketName("destbucketname");
request.setDestinationObjectKey("destobjectname");
obsClient.copyObject(request);
```

说明

- 复制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服务端错误码**。

10.6 恢复多版本归档存储对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.restoreObject接口传入版本号（versionId）来恢复多版本归档存储对象。

本示例用于通过设置versionId值将destbucketname桶中destobjectname的多版本归档存储对象，快速恢复为标准存储对象。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

RestoreObjectRequest request = new RestoreObjectRequest("bucketname", "objectname", 1);
// 使用快速恢复方式，恢复多版本对象
request.setRestoreTier(RestoreTierEnum.EXPEDITED);
request.setVersionId("versionid");
obsClient.restoreObject(request);
```

说明

- 恢复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服务端错误码**。

10.7 列举多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.listVersions列举多版本对象。

该接口可设置的参数如下：

参数	作用
bucketName	桶名。
prefix	限定返回的对象名必须带有prefix前缀。
keyMarker	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。
maxKeys	列举多版本对象的最大数目，取值范围为1~1000，当超出范围时，按照默认的1000进行处理。
delimiter	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。

参数	作用
versionIdMarker	与keyMarker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。

说明

- 如果versionIdMarker不是keyMarker的一个版本号，则该参数无效。
- ObsClient.listVersions返回结果包含多版本对象和对象删除标记。

简单列举

以下代码展示如何简单列举多版本对象，最多返回1000个对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListVersionsResult result = obsClient.listVersions("bucketname");

for(VersionOrDeleteMarker v : result.getVersions()){
    Log.i("ListVersions", "\t" + v.getKey());
    Log.i("ListVersions","\t" + v.getOwner());
    Log.i("ListVersions","\t" + v.isDeleteMarker());
}
```

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ListVersionsResult.isTruncated为true表明本次没有返回全部对象，并可通过ListVersionsResult.getNextKeyMarker和ListVersionsResult.getNextVersionIdMarker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。

指定数目列举

以下代码展示如何指定数目列举多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListVersionsResult result = obsClient.listVersions("bucketname", 100);
for(VersionOrDeleteMarker v : result.getVersions()){
    Log.i("ListVersions","\t" + v.getKey());
    Log.i("ListVersions","\t" + v.getOwner());
```



```
    Log.i("ListVersions","\t" + v.isDeleteMarker());
}
```

指定前缀列举

以下代码展示如何指定前缀列举多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 列举带有prefix前缀的100个多版本对象
ListVersionsRequest request = new ListVersionsRequest ("bucketname", 100);
request.setPrefix("prefix");
ListVersionsResult result = obsClient.listVersions(request);
for(VersionOrDeleteMarker v : result.getVersions()){
    Log.i("ListVersions","\t" + v.getKey());
    Log.i("ListVersions","\t" + v.getOwner());
    Log.i("ListVersions","\t" + v.isDeleteMarker());
}
```

指定起始位置列举

以下代码展示如何指定起始位置列举多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 列举对象名字典序在"test"之后的100个多版本对象
ListVersionsRequest request = new ListVersionsRequest ("bucketname", 100);
request.setKeyMarker("test");
ListVersionsResult result = obsClient.listVersions(request);

for(VersionOrDeleteMarker v : result.getVersions()){
    Log.i("ListVersions","\t" + v.getKey());
    Log.i("ListVersions","\t" + v.getOwner());
    Log.i("ListVersions","\t" + v.isDeleteMarker());
}
```

分页列举全部多版本对象

以下代码展示分页列举全部多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
```



```
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListVersionsResult result;
ListVersionsRequest request = new ListVersionsRequest ("bucketname", 100);
do{
    result = obsClient.listVersions(request);
    for(VersionOrDeleteMarker v : result.getVersions()){
        Log.i("ListVersions","\t" + v.getKey());
        Log.i("ListVersions","\t" + v.getOwner());
        Log.i("ListVersions","\t" + v.isDeleteMarker());
    }
    request.setKeyMarker(result.getNextKeyMarker());
    request.setVersionIdMarker(result.getNextVersionIdMarker());
}while(result.isTruncated());
```

列举文件夹中的所有多版本对象

OBS本身是没有文件夹的概念的，桶中存储的元素只有对象。文件夹对象实际上是一个大小为0且对象名以“/”结尾的对象，将这个文件夹对象名作为前缀，即可模拟列举文件夹中对象的功能。以下代码展示如何列举文件夹中的多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

ListVersionsResult result;
ListVersionsRequest request = new ListVersionsRequest ("bucketname", 100);
// 设置文件夹对象名"dir/"为前缀
request.setPrefix("dir/");
do{
    result = obsClient.listVersions(request);
    for(VersionOrDeleteMarker v : result.getVersions()){
        Log.i("ListVersions","\t" + v.getKey());
        Log.i("ListVersions","\t" + v.getOwner());
        Log.i("ListVersions","\t" + v.isDeleteMarker());
    }
    request.setKeyMarker(result.getNextKeyMarker());
    request.setVersionIdMarker(result.getNextVersionIdMarker());
}while(result.isTruncated());
```

按文件夹分组列举所有多版本对象

以下代码展示如何按文件夹分组，列举桶内所有多版本对象：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
ListVersionsRequest request = new ListVersionsRequest ("bucketname", 1000);
request.setDelimiter("/");
ListVersionsResult result = obsClient.listVersions(request);
Log.i("ListVersions", "Objects in the root directory:");
```

```
for(VersionOrDeleteMarker v : result.getVersions()){
    Log.i("ListVersions","\t" + v.getKey());
    Log.i("ListVersions","\t" + v.getOwner());
    Log.i("ListVersions","\t" + v.isDeleteMarker());
}

listVersionsByPrefix(obsClient, result);
```

递归列出子文件夹中多版本对象的函数*listVersionsByPrefix*的示例代码如下：

```
static void listVersionsByPrefix(ObsClient obsClient, ListVersionsResult result) throws ObsException{
    for(String prefix : result.getCommonPrefixes()){
        Log.i("ListVersions", "Objects in folder [" + prefix + "]:");
        ListVersionsRequest request = new ListVersionsRequest ("bucketname", 1000);
        request.setDelimiter("/");
        request.setPrefix(prefix)
        result = obsClient.listVersions(request);
        for(VersionOrDeleteMarker v : result.getVersions()){
            Log.i("ListVersions","\t" + v.getKey());
            Log.i("ListVersions","\t" + v.getOwner());
            Log.i("ListVersions","\t" + v.isDeleteMarker());
        }
        listVersionsByPrefix(obsClient, result);
    }
}
```

说明

- 以上代码示例没有考虑文件夹中多版本对象数超过1000个的情况。
- 由于是需要列举出文件夹中的对象和子文件夹，且文件夹对象总是以“/”结尾，因此delimiter总是为“/”。
- 每次递归的返回结果中ListVersionsResult.getVersions包含的是文件夹中的多版本对象；ListVersionsResult.getCommonPrefixes包含的是文件夹的子文件夹。

10.8 多版本对象权限

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

设置多版本对象访问权限

您可以通过ObsClient.setObjectAcl接口传入版本号（versionId）设置多版本对象的访问权限，示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 通过预定义访问策略设置多版本对象访问权限为公共读
obsClient.setObjectAcl("bucketname", "objectname", AccessControlList.REST_CANNED_PUBLIC_READ, "versionid");
```

```
AccessControlList acl = new AccessControlList();
Owner owner = new Owner();
owner.setId("ownerid");
acl.setOwner(owner);
// 为所有用户设置读权限
acl.grantPermission(GroupGrantee.ALL_USERS, Permission.PERMISSION_READ);
// 直接设置多版本对象访问权限
obsClient.setObjectAcl("bucketname", "objectname", acl, "versionid");
```

说明

ACL中需要填写的所有者（Owner）或者被授权用户（Grantee）的ID，是指用户的账号ID，可通过OBS控制台“我的凭证”页面查看。

获取多版本对象访问权限

您可以通过ObsClient.getObjectAcl接口传入版本号（versionId）获取多版本对象的访问权限，示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

AccessControlList acl = obsClient.getObjectAcl("bucketname", "objectname", "versionid");
System.out.println(acl);
```

10.9 删除多版本对象

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

删除单个多版本对象

您可以通过ObsClient.deleteObject接口传入版本号（versionId）删除多版本对象。

本示例用于通过设置多版本对象的VersionId值，删除桶名为“bucketname”里，名称为“objectname”的对象。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
```



```
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
obsClient.deleteObject("bucketname", "objectname", "versionid");
```

批量删除多版本对象

您可以通过ObsClient.deleteObjects接口传入每个待删除对象的版本号（versionId）批量删除多版本对象。

本示例用于通过设置多版本对象的VersionId值，批量删除桶名为“bucketname”里，名称为“objectname1”，“objectname2”和“objectname3”的对象。

代码示例如下所示：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

DeleteObjectsRequest request = new DeleteObjectsRequest("bucketname");
request.setQuiet(false);
List<KeyAndVersion> toDelete = new ArrayList<KeyAndVersion>();
toDelete.add(new KeyAndVersion("objectname1", "versionid1"));
toDelete.add(new KeyAndVersion("objectname2", "versionid2"));
toDelete.add(new KeyAndVersion("objectname3", "versionid3"));
request.setKeyAndVersions(toDelete.toArray(new KeyAndVersion[toDelete.size()]));
DeleteObjectsResult result = obsClient.deleteObjects(request);

Log.i("DeleteObjects","\t" + result.getDeletedObjectResults());
Log.i("DeleteObjects","\t" + result.getErrorResults());
```


11 生命周期管理

11.1 生命周期管理简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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型、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型。
  b. 直接指定满足前缀的对象转换为指定的存储类型的日期。
- 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型。
- 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。
- 桶的多版本状态必须处于Enabled或者Suspended，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。

11.2 设置生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketLifecycle设置桶的生命周期规则。

设置对象转换策略

以下代码展示了如何设置最新版本对象和历史版本对象的转换策略：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

LifecycleConfiguration config = new LifecycleConfiguration();
LifecycleConfiguration.Rule rule = config.new Rule();
rule.setEnabled(true);
rule.setId("rule1");
rule.setPrefix("prefix");
LifecycleConfiguration.Transition transition = config.new Transition();
// 指定满足前缀的对象创建30天后转换
transition.setDays(30);
// 指定对象转换后的存储类型
transition.setObjectStorageClass(StorageClassEnum.STANDARD);
// 直接指定满足前缀的对象转换日期
// transition.setDate(new SimpleDateFormat("yyyy-MM-dd").parse("2018-10-31"));
rule.getTransitions().add(transition);

LifecycleConfiguration.NoncurrentVersionTransition noncurrentVersionTransition = config.new NoncurrentVersionTransition();
// 指定满足前缀的对象成为历史版本30天后转换
noncurrentVersionTransition.setDays(30);
// 指定历史版本对象转换后的存储类型
noncurrentVersionTransition.setObjectStorageClass(StorageClassEnum.COLD);
rule.getNoncurrentVersionTransitions().add(noncurrentVersionTransition);

config.addRule(rule);

obsClient.setBucketLifecycle("bucketname", config);
```

设置对象过期时间

以下代码展示了如何设置最新版本对象和历史版本对象的过期时间：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

LifecycleConfiguration config = new LifecycleConfiguration();

Rule rule = config.new Rule();
rule.setEnabled(true);
rule.setId("rule1");
rule.setPrefix("prefix");
Expiration expiration = config.new Expiration();
// 指定满足前缀的对象创建60天后过期
expiration.setDays(60);
// 直接指定满足前缀的对象过期时间
// expiration.setDate(new SimpleDateFormat("yyyy-MM-dd").parse("2018-12-31"));
rule.setExpiration(expiration);

NoncurrentVersionExpiration noncurrentVersionExpiration = config.new NoncurrentVersionExpiration();
// 指定满足前缀的对象成为历史版本60天后过期
noncurrentVersionExpiration.setDays(60);
rule.setNoncurrentVersionExpiration(noncurrentVersionExpiration);
config.addRule(rule);

obsClient.setBucketLifecycle("bucketname", config);
```

11.3 查看生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketLifecycle查看桶的生命周期规则。

本示例用于查看桶名为“bucketname”的生命周期规则。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

LifecycleConfiguration config = obsClient.getBucketLifecycle("bucketname");
```

```
for (LifecycleConfiguration.Rule rule : config.getRules()) {
    Log.i("GetBucketLifecycleConfiguration",rule.getId());
    Log.i("GetBucketLifecycleConfiguration",rule.getPrefix());
    for(LifecycleConfiguration.Transition transition : rule.getTransitions()){
        Log.i("GetBucketLifecycleConfiguration",String.valueOf(transition.getDays()));
        Log.i("GetBucketLifecycleConfiguration",transition.getStorageClass());
    }
    Log.i("GetBucketLifecycleConfiguration",rule.getExpiration() != null ?
String.valueOf(rule.getExpiration().getDays()) : "");
    for(LifecycleConfiguration.NoncurrentVersionTransition noncurrentVersionTransition :
rule.getNoncurrentVersionTransitions()){
        Log.i("GetBucketLifecycleConfiguration",String.valueOf(noncurrentVersionTransition.getDays()));
        Log.i("GetBucketLifecycleConfiguration",noncurrentVersionTransition.getStorageClass());
    }
    Log.i("GetBucketLifecycleConfiguration",rule.getNoncurrentVersionExpiration() != null ?
String.valueOf(rule.getNoncurrentVersionExpiration().getDays()) : "");
}
```

说明

- 查看生命周期规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

11.4 删除生命周期规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.deleteBucketLifecycle删除桶的生命周期规则。

本示例用于删除桶名为“bucketname”的生命周期规则。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

obsClient.deleteBucketLifecycle("bucketname");
```

说明

- 删除生命周期规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

12 跨域资源共享

12.1 跨域资源共享简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

跨域是指不同域名之间相互访问。跨域访问是浏览器出于安全考虑而设置的一个限制，即同源策略。

由于JavaScript同源策略的限制，A域名下的JavaScript无法操作B域名或C域名下的对象。

同协议、同域名（或IP）、以及同端口视为同一个域。两个页面的协议、域名和端口（如果指定了端口）相同，则视为同源。

跨域资源共享（CORS）允许Web端的应用程序访问不属于本域的资源。OBS提供接口方便开发者控制跨域访问的权限。

更多关于跨域资源共享的内容请参考**跨域资源访问**。

12.2 设置跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketCors设置桶的跨域规则，如果原规则存在则覆盖原规则。

本示例用于设置桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketCors cors = new BucketCors();

List<BucketCorsRule> rules = new ArrayList<BucketCorsRule>();
BucketCorsRule rule = new BucketCorsRule();

ArrayList<String> allowedOrigin = new ArrayList<String>();
// 指定允许跨域请求的来源
allowedOrigin.add( "http://www.a.com");
allowedOrigin.add( "http://www.b.com");
rule.setAllowedOrigin(allowedOrigin);

ArrayList<String> allowedMethod = new ArrayList<String>();
// 指定允许的跨域请求方法(GET/PUT/DELETE/POST/HEAD)
allowedMethod.add("GET");
allowedMethod.add("HEAD");
allowedMethod.add("PUT");
rule.setAllowedMethod(allowedMethod);

ArrayList<String> allowedHeader = new ArrayList<String>();
// 控制在OPTIONS预取指令中Access-Control-Request-Headers头中指定的header是否被允许使用
allowedHeader.add("x-obs-header");
rule.setAllowedHeader(allowedHeader);

ArrayList<String> exposeHeader = new ArrayList<String>();
// 指定允许用户从应用程序中访问的header
exposeHeader.add("x-obs-expose-header");
rule.setExposeHeader(exposeHeader);

// 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
rule.setMaxAgeSecond(10);
rules.add(rule);
cors.setRules(rules);

obsClient.setBucketCors("bucketname", cors);
```

说明

- AllowedOrigins、AllowedHeaders都能够最多支持一个“*”通配符。“*”表示对于所有的域来源或者头域都满足。

12.3 查看跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketCors查看桶的跨域规则。

本示例用于查看桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);


BucketCors cors = obsClient.getBucketCors("bucketname");
for(BucketCorsRule rule : cors.getRules()){
    Log.i("GetBucketCors","\t" + rule.getId());
    Log.i("GetBucketCors","\t" + rule.getMaxAgeSecond());
    Log.i("GetBucketCors","\t" + rule.getAllowedHeader());
    Log.i("GetBucketCors","\t" + rule.getAllowedOrigin());
    Log.i("GetBucketCors","\t" + rule.getAllowedMethod());
    Log.i("GetBucketCors","\t" + rule.getExposeHeader());
}
```

说明

- 查看跨域规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

12.4 删除跨域规则

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.deleteBucketCors删除桶的跨域规则。

本示例用于删除桶名为“bucketname”的跨域规则。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

obsClient.deleteBucketCors("bucketname");
```

说明

- 删除跨域规则过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

13 设置访问日志

13.1 日志简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS允许您对桶设置访问日志记录，设置之后对于桶的访问会被记录成日志，日志存储在OBS上您指定的目标桶中。

出于分析或审计等目的，用户可以开启日志记录功能。通过访问日志记录，桶的拥有者可以深入分析访问该桶的用户请求性质、类型或趋势。

当用户开启一个桶的日志记录功能后，OBS会自动对这个桶的访问请求记录日志，并生成日志文件写入用户指定的桶（即目标桶）中。

日志文件存放位置需要在开启桶日志功能时指定，可以存放到您拥有的，且与开启日志功能的桶位于同一区域的任一存储桶，当然也包括开启日志功能的桶本身。

更多关于访问日志的内容请参考**日志记录**。

13.2 开启桶日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketLogging开启桶日志功能。

须知

日志目标桶与源桶必须在同一个区域（region）。

说明

如果桶的存储类型为低频访问存储或归档存储，则不能作为日志目标桶。

开启桶日志

以下代码展示了如何开启桶日志：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketLoggingConfiguration config = new BucketLoggingConfiguration();
// 设置委托名字，需要到统一身份认证服务（IAM）创建委托
config.setAgency("your agency");
config.setTargetBucketName("targetbucketname");
config.setLogfilePrefix("targetprefix");

obsClient.setBucketLogging("bucketname", config);
```

为日志对象设置权限

以下代码展示了如何为日志对象设置权限：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

String targetBucket = "targetbucketname";

// 设置桶的日志配置
BucketLoggingConfiguration config = new BucketLoggingConfiguration();
// 设置委托名字，需要到统一身份认证服务（IAM）创建委托
config.setAgency("your agency");
config.setTargetBucketName(targetBucket);
config.setLogfilePrefix("prefix");

// 为所有用户设置对日志对象的读权限
GrantAndPermission grant1 = new GrantAndPermission(GroupGrantee.ALL_USERS,
Permission.PERMISSION_READ);
config.setTargetGrants(new GrantAndPermission[]{grant1});

obsClient.setBucketLogging("bucketname", config);
```

13.3 查看桶日志配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketLogging查看桶日志配置。

本示例用于查看桶名为“bucketname”的日志配置。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketLoggingConfiguration config = obsClient.getBucketLogging("bucketname");
Log.i("GetBucketLoggingConfiguration","\t" + config.getTargetBucketName());
Log.i("GetBucketLoggingConfiguration","\t" + config.getLogfilePrefix());
```

说明

- 查看桶日志过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

13.4 关闭桶日志

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

关闭桶日志功能实际上就是调用ObsClient.setBucketLogging将日志配置清空。

本示例用于关闭桶名为“bucketname”的日志配置。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```

```
// 对桶设置空的日志配置
obsClient.setBucketLogging("bucketname", new BucketLoggingConfiguration());
```

说明

- 关闭桶日志过程中返回的错误码含义、问题原因及处理措施可参考**OBS服务端错误码**。

14 静态网站托管

14.1 静态网站托管简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

静态网站通常仅包含静态网页，以及可能包含部分可在客户端运行的脚本，如JavaScript、Flash等。相比之下，动态网站则依赖于服务器端处理脚本，包括PHP、JSP或ASP.Net等。

您可以将静态网站文件上传至OBS的桶中作为对象，并对这些对象赋予公共读权限，然后将该桶配置成静态网站托管模式，以实现在OBS上托管静态网站的目的。

第三方用户在访问您网站的时候，实际上是在访问OBS的桶中的对象。

在使用静态网站托管功能时，OBS还支持配置请求重定向，通过重定向配置您可以将特定的请求或所有请求实施重定向。

更多关于静态网站托管的内容请参考**静态网站托管**。

14.2 网站文件托管

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

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

以下代码展示了如何实现网站文件托管：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 上传对象，设置对象MIME类型
PutObjectRequest request = new PutObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("test.html");
request.setFile(new File("localfile.html"));
ObjectMetadata metadata = new ObjectMetadata();
metadata.setContentType("text/html");
request.setMetadata(metadata);
obsClient.putObject(request);

// 设置对象访问权限为公共读
obsClient.setObjectAcl("bucketname", "test.html", AccessControlList.REST_CANNED_PUBLIC_READ);
```

说明

上例中您可以使用https://bucketname.your-endpoint/test.html在浏览器直接访问托管的文件。

14.3 设置托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketWebsite设置桶的托管配置。

配置默认主页和错误页面

以下代码展示了如何配置默认主页和错误页面：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```



```
WebsiteConfiguration config = new WebsiteConfiguration();
// 配置默认主页
config.setSuffix("index.html");
// 配置错误页面
config.setKey("error.html");
obsClient.setBucketWebsite("bucketname", config);
```

配置重定向规则

以下代码展示了如何配置重定向规则：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

WebsiteConfiguration config = new WebsiteConfiguration();
// 配置默认主页
config.setSuffix("index.html");
// 配置错误页面
config.setKey("error.html");

RouteRule rule = new RouteRule();
Redirect r = new Redirect();
r.setHostName("www.example.com");
r.setHttpRedirectCode("305");
r.setRedirectProtocol(ProtocolEnum.HTTP);
r.setReplaceKeyPrefixWith("replacekeyprefix");
rule.setRedirect(r);
RouteRuleCondition condition = new RouteRuleCondition();
condition.setHttpErrorCodeReturnedEquals("404");
condition.setKeyPrefixEquals("keyprefix");
rule.setCondition(condition);
config.getRouteRules().add(rule);

obsClient.setBucketWebsite("bucketname", config);
```

配置所有请求重定向

以下代码展示了如何配置所有请求重定向：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

WebsiteConfiguration config = new WebsiteConfiguration();
RedirectAllRequest redirectAll = new RedirectAllRequest();
redirectAll.setHostName("www.example.com");
// 支持http和https
redirectAll.setRedirectProtocol(ProtocolEnum.HTTP);
config.setRedirectAllRequestsTo(redirectAll);

obsClient.setBucketWebsite("bucketname", config);
```


14.4 查看托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketWebsite查看桶的托管配置。

本示例用于查看桶名为“bucketname”的托管配置。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

WebsiteConfiguration config = obsClient.getBucketWebsite("bucketname");
Log.i("GetBucketWebsiteConfiguration", "\t" + config.getKey());
Log.i("GetBucketWebsiteConfiguration", "\t" + config.getSuffix());
for(RouteRule rule : config.getRouteRules()){
    Log.i("GetBucketWebsiteConfiguration", "\t" + rule);
}
```

说明

- 查看桶托管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服务端错误码**。

14.5 清除托管配置

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.deleteBucketWebsite清除桶的托管配置。

本示例用于清除桶名为“bucketname”的托管配置。

代码示例如下所示:

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
```

```
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

obsClient.deleteBucketWebsite("bucketname");
```

说明

- 清除桶托管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服务端错误码**。

15 标签管理

15.1 标签简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

标签用于标识OBS中的桶，以此来达到对OBS中的桶进行分类的目的。

15.2 设置桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.setBucketTagging设置桶标签。以下代码展示了如何设置桶标签：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";

// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketTagInfo bucketTagInfo = new BucketTagInfo();

TagSet tagSet = new TagSet();
```

```
tagSet.addTag("tag1", "value1");
tagSet.addTag("tag2", "value2");

bucketTagInfo.setTagSet(tagSet);
obsClient.setBucketTagging("bucketname", bucketTagInfo);
```

说明

- 每个桶支持最多10个标签。
- 标签的Key和Value支持Unicode。

15.3 查看桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.getBucketTagging查看桶标签。以下代码展示了如何查看桶标签：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

BucketTagInfo bucketTagInfo = obsClient.getBucketTagging("bucketname");
for(Tag tag : bucketTagInfo.getTagSet().getTags()){
    Log.i("GetBucketTagging", "\t" + tag.getKey() + ":" + tag.getValue());
}
```

15.4 删除桶标签

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

您可以通过ObsClient.deleteBucketTagging删除桶标签。以下代码展示了如何删除桶标签：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
```



```
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

obsClient.deleteBucketTagging("bucketname");
```


16 服务端加密

16.1 服务端加密简介

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS支持服务端加密功能，使对象加密的行为在OBS服务端进行。

更多关于服务端加密的内容请参考**服务端加密**。

16.2 加密说明

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

OBS Android SDK支持服务端加密的接口见下表：

OBS Android SDK接口方法	描述	支持加密类型
ObsClient.putObject	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
ObsClient.getObject	下载对象时设置解密算法、密钥，用于解密对象。	SSE-C

OBS Android SDK接口方法	描述	支持加密类型
ObsClient.copyObject	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
ObsClient.getObjectMetadata	获取对象元数据时设置解密算法、密钥，用于解密对象。	SSE-C
ObsClient.initiateMultipartUpload	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
ObsClient.uploadPart	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
ObsClient.copyPart	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

16.3 加密示例

须知

开发过程中，您有任何问题可以在github上**提交issue**。**接口参考文档**详细介绍了每个接口的参数和使用方法。

上传对象加密

以下代码展示了在上传对象时使用服务端加密功能：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

PutObjectRequest request = new PutObjectRequest();
request.setBucketName("bucketname");
request.setObjectKey("objectname");
request.setFile(new File("localfile"));
// 设置SSE-C算法加密对象
SseCHeader sseCHeader = new SseCHeader();
sseCHeader.setAlgorithm(ServerAlgorithm.AES256);
// 设置SSE-C方式下使用的密钥，用于加解密对象，该值是密钥进行base64encode后的值
```

```
sseCHeader.setSseCKeyBase64("your base64 sse-c key generated by AES-256 algorithm");

request.setSseCHeader(sseCHeader);
obsClient.putObject(request);
```

下载对象解密

以下代码展示了在下载对象时使用服务端解密功能：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);

GetObjectRequest request = new GetObjectRequest("bucketname", "objectname");
// 设置SSE-C算法解密对象
SseCHeader sseCHeader = new SseCHeader();
sseCHeader.setAlgorithm(ServerAlgorithm.AES256);
// 此处的密钥必须和上传对象加密时使用的密钥一致
sseCHeader.setSseCKeyBase64("your base64 sse-c key generated by AES-256 algorithm");

request.setSseCHeader(sseCHeader);
ObsObject obsObject = obsClient.getObject(request);

obsObject.getObjectContent().close();
```


17 异常处理

17.1 OBS 服务端错误码

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：

错误码	描述	HTTP状态码
AccessDenied	拒绝访问。	403 Forbidden
AccessForbidden	权限不足。	403 Forbidden
AccountProblem	用户的账户出现异常（过期、冻结等），不能成功地完成操作。	403 Forbidden
AllAccessDisabled	用户无权限执行某操作。	403 Forbidden
AmbiguousGrantByEmailAddress	用户提供的Email地址关联的账户超过了1个。	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。	400 Bad Request
BadDomainName	域名不合法。	400 Bad Request
BadRequest	请求参数不合法。	400 Bad Request
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。	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。	409 Conflict

错误码	描述	HTTP状态码
BucketNotEmpty	用户尝试删除的桶不为空。	409 Conflict
CredentialsNotSupported	该请求不支持证书验证。	400 Bad Request
CustomDomainAreadyExist	配置了已存在的域。	400 Bad Request
CustomDomainNotExist	操作的域不存在。	400 Bad Request
DeregisterUserId	用户已经注销。	403 Forbidden
EntityTooSmall	用户试图上传的对象大小小于系统允许的最小大小。	400 Bad Request
EntityTooLarge	用户试图上传的对象大小超过了系统允许的最大大小。	400 Bad Request
FrozenUserId	用户被冻结。	403 Forbidden
IllegalVersioningConfigurationException	请求中的版本配置无效。	400 Bad Request
IllegalLocationConstraintException	配置了与所在Region不匹配的区域限制。	400 Bad Request
InArrearOrInsufficientBalance	因为ACL而没有权限进行某种操作。	403 Forbidden
IncompleteBody	请求体不完整。	400 Bad Request
IncorrectNumberOfFilesInPostRequest	每个POST请求都需要带一个上传的文件。	400 Bad Request
InlineDataTooLarge	Inline Data超过了允许的最大长度。	400 Bad Request
InsufficientStorageSpace	存储空间不足。	403 Forbidden
InternalError	系统遇到内部错误，请重试。	500 Internal Server Error
InvalidAccessKeyId	系统记录中不存在客户提供的Access Key Id。	403 Forbidden
InvalidAddressingHeader	用户必须指定匿名角色。	N/A
InvalidArgument	无效的参数。	400 Bad Request
InvalidBucketName	请求中指定的桶名无效。	400 Bad Request
InvalidBucket	请求访问的桶已不存在。	400 Bad Request

错误码	描述	HTTP状态码
InvalidBucketState	无效的桶状态。	409 Conflict
InvalidBucketStoragePolicy	修改桶策略时，提供的新策略不合法。	400 Bad Request
InvalidDigest	HTTP头中指定的Content-MD5值无效。	400 Bad Request
InvalidEncryptionAlgorithmError	错误的加密算法。	400 Bad Request
InvalidLocationConstraint	创建桶时，指定的location不合法。	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。	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。	400 Bad Request
InvalidPayer	所有对这个对象的访问已经无效了。	403 Forbidden
InvalidPolicyDocument	表单中的内容与策略文档中指定的条件不一致。	400 Bad Request
InvalidRange	请求的range不可获得。	416 Client Requested Range Not Satisfiable
InvalidRedirectLocation	无效的重定向地址。	400 Bad Request
InvalidRequest	无效请求。	400 Bad Request
InvalidRequestBody	POST请求体无效。	400 Bad Request
InvalidSecurity	提供的安全证书无效。	403 Forbidden
InvalidStorageClass	用户指定的Storage Class无效。	400 Bad Request
InvalidTargetBucketForLogging	delivery group对目标桶无ACL权限。	400 Bad Request
InvalidURI	无法解析指定的URI。	400 Bad Request
KeyTooLong	提供的Key过长。	400 Bad Request

错误码	描述	HTTP状态码
MalformedACLError	提供的XML格式错误，或者不符合要求的格式。	400 Bad Request
MalformedError	请求中携带的XML格式不正确。	400 Bad Request
MalformedLoggingStatus	Logging的XML格式不正确。	400 Bad Request
MalformedPolicy	Bucket policy检查不通过。	400 Bad Request
MalformedPOSTRequest	POST请求的请求体不是结构化良好的多段或形式化数据。	400 Bad Request
MalformedQuotaError	Quota的XML格式不正确。	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。错误消息是：“The XML you provided was not well-formed or did not validate against our published schema.”。	400 Bad Request
MaxMessageLengthExceeded	请求消息过长。	400 Bad Request
MaxPostPreDataLengthExceededError	在上传文件前面的POST请求域过大。	400 Bad Request
MetadataTooLarge	元数据消息头超过了允许的最大元数据大小。	400 Bad Request
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.	405 Method Not Allowed
MissingContentLength	必须要提供HTTP消息头中的Content-Length字段。	411 Length Required
MissingRegion	请求中缺少Region信息，且系统无默认Region。	400 Bad Request
MissingRequestBodyError	当用户发送一个空的XML文档作为请求时会发生。错误消息是：“Request body is empty.”。	400 Bad Request

错误码	描述	HTTP状态码
MissingRequiredHeader	请求中缺少必要的头域。	400 Bad Request
MissingSecurityHeader	请求缺少一个必须的头。	400 Bad Request
NoSuchBucket	指定的桶不存在。	404 Not Found
NoSuchBucketPolicy	桶policy不存在。	404 Not Found
NoSuchCORSConfiguration	CORS配置不存在。	404 Not Found
NoSuchCustomDomain	请求的用户域不存在。	404 Not Found
NoSuchKey	指定的Key不存在。	404 Not Found
NoSuchLifecycleConfiguration	请求的LifeCycle不存在。	404 Not Found
NoSuchPolicy	给定的policy名字不存在。	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。	404 Not Found
NoSuchVersion	请求中指定的version ID与现存的所有版本都不匹配。	404 Not Found
NoSuchWebsiteConfiguration	请求的Website不存在。	404 Not Found
NotImplemented	用户提供的消息头功能上还没有实现。	501 Not Implemented
NotSignedUp	账户未在系统中注册，必须先在系统中注册了才能使用该账户。	403 Forbidden
OperationAborted	另外一个冲突的操作当前正作用在这个资源上，请重试。	409 Conflict
PermanentRedirect	尝试访问的桶必须使用指定的节点，请将以后的请求发送到这个节点。	301 Moved Permanently
PreconditionFailed	用户指定的先决条件中至少有一项没有包含。	412 Precondition Failed
Redirect	临时重定向。	307 Moved Temporarily
RequestIsNotMultiPartContent	桶POST必须是闭式的多段/表单数据。	400 Bad Request

错误码	描述	HTTP状态码
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。	400 Bad Request
RequestTimeTooSkewed	请求的时间与服务器的时间相差太大。	403 Forbidden
RequestTorrentOfBucketError	不允许请求桶的torrent文件。	400 Bad Request
ServiceNotImplemented	请求的方法服务端没有实现。	501 Not Implemented
ServiceNotSupported	请求的方法服务端不支持。	409 Conflict
ServiceUnavailable	服务器过载或者内部错误异常。	503 Service Unavailable
SignatureDoesNotMatch	请求中带的签名与系统计算得到的签名不一致。检查您的访问密钥（AK和SK）和签名计算方法。	403 Forbidden
SlowDown	请降低请求频率。	503 Service Unavailable
System Capacity Not enough	系统空间不足异常。	403 Forbidden
TooManyCustomDomains	配置了过多的用户域。	400 Bad Request
TemporaryRedirect	当DNS更新时，请求将被重定向到桶。	307 Moved Temporarily
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。	400 Bad Request
TooManyObjectCopied	用户单个对象被拷贝的数量超过系统上限。	400 Bad Request
TooManyWrongSignature	因高频错误请求被拒绝服务。	400 Bad Request
UnexpectedContent	该请求不支持带内容字段。	400 Bad Request
UnresolvableGrantByEmailAddress	用户提供的Email与记录中任何账户的都不匹配。	400 Bad Request
UserKeyMustBeSpecified	请求中缺少用户的AK信息。	400 Bad Request
WebsiteRedirect	Website请求缺少bucketName。	301 Moved Permanently

错误码	描述	HTTP状态码
KMS.DisabledException	SSE-KMS加密方式下，主密钥被禁用。	400 Bad Request
KMS.NotFoundException	SSE-KMS加密方式下，主密钥不存在。	400 Bad Request
RestoreAlreadyInProgress	对象正在恢复，请求冲突。	409 Conflict
ObjectHasAlreadyRestored	已经恢复的对象，禁止缩短恢复保存时间。	409 Conflict
InvalidObjectState	恢复对象不是归档存储对象。	403 Forbidden
InvalidTagError	配置桶标签时，提供了无效的Tag。	400 Bad Request
NoSuchTagSet	指定的桶没有设置标签。	404 Not Found

17.2 SDK 自定义异常

SDK自定义异常（ObsException）是由ObsClient统一抛出的异常，继承自java.lang.RuntimeException类。通常是OBS服务端错误，包含**OBS错误码**、错误信息等，便于用户定位问题，并做出适当的处理。

ObsException通常包含以下错误信息：

- ObsException.getResponseCode：HTTP状态码。
- ObsException.getErrorCode：OBS服务端错误码。
- ObsException.getErrorMessage：OBS服务端错误描述。
- ObsException.getErrorRequestId：OBS服务端返回的请求ID。
- ObsException.getErrorHostId：请求的服务端ID。
- ObsException.getResponseHeaders：HTTP响应头信息。

17.3 SDK 公共响应头

调用ObsClient类的相关接口成功后，均会返回公共响应头类，即HeaderResponse类实例（或其子类实例），该类包含了HTTP/HTTPS的响应头信息。

处理公共响应头的示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 创建ObsClient实例
ObsClient obsClient = new ObsClient(ak, sk, endPoint);
```

```
HeaderResponse response = obsClient.createBucket("bucketname");

//从公共响应头中获取request-id
Log.i("CreateBucket", "\t" + response.getRequestId());

obsClient.close();
```

17.4 日志分析

日志路径

OBS Android SDK的日志路径是通过LogConfigurator.setLogFileDir指定的，默认存放于SD卡的logs目录下。日志文件名为：OBS-SDK.log。

日志内容格式

SDK日志格式为：日志时间|线程号|日志级别|日志内容。示例如下：

```
2017-08-21 17:40:07 133|main|INFO |HttpClient cost 157 ms to apply http request
2017-08-21 17:40:07 133|main|INFO |Received expected response code: true
2017-08-21 17:40:07 133|main|INFO |expected code(s): [200, 204].

2017-08-21 17:40:06 820|main|INFO |Storage|1|HTTP+XML|ObsClient||||2017-08-21 17:40:05|2017-08-21 17:40:06|||0|
2017-08-21 17:40:07 136|main|INFO |Storage|1|HTTP+XML|setObjectAcl||||2017-08-21 17:40:06|2017-08-21 17:40:07|||0|
2017-08-21 17:40:07 137|main|INFO |ObsClient [setObjectAcl] cost 312 ms
```

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TRACE日志信息最丰富，ERROR日志信息最少。可以通过LogConfigurator.setLogLevel设置日志级别。

具体说明见下表：

日志级别	说明	OBS Android SDK对应值
OFF	关闭级别，如果设置为这个级别，日志打印功能将被关闭。	LogConfigurator.OFF
TRACE	跟踪级别，如果设置为这个级别，将打印所有日志信息。通常不建议使用。	LogConfigurator.TRACE
DEBUG	调试级别，如果设置为这个级别，除了打印INFO级别以上的信息外，还将打印每次HTTP/HTTPS请求和响应的头信息，鉴权算法计算出的StringToSign信息等。	LogConfigurator.DEBUG
INFO	信息级别，如果设置为这个级别，除了打印WARN级别以上的信息外，还将打印HTTP/HTTPS请求的耗时时间，ObsClient接口的耗时时间等。	LogConfigurator.INFO

日志级别	说明	OBS Android SDK对应值
WARN	告警级别，如果设置为这个级别，除了打印ERROR级别以上的信息外，还将打印一些关键事件的信息，如重试请求超过最大次数等。	LogConfigurator.WARN
ERROR	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。	LogConfigurator.ERROR

日志无法生成

如发现无法生成日志，检查AndroidManifest.xml文件中是否加入了以下权限：

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

17.5 缺少类错误

使用 OBS Android SDK 进行二次开发时如果报缺少类错误，请检查libs目录下除esdk-obs-java-x.x.x.jar外是否包含以下依赖库：

- java-xmlbuilder-1.3.jar
- okhttp-4.8.0.jar
- okio-2.7.0.jar

17.6 连接超时异常

如果通过ObsException.getResponseCode获取到的错误码为408，表明连接OBS服务超时，出现这类异常的原因一般是服务地址（Endpoint）错误或网络不通导致无法连接OBS服务，此时请检查服务地址和网络状况。

17.7 资源无法释放

如果发现使用OBS Android SDK后存在内存泄露或OBS服务端连接未断开等情况，请检查是否正确调用了ObsClient.close以及ObsObject.getObjectContent.close释放资源。

17.8 签名不匹配异常

如果从ObsException中获取到HTTP状态码为403，OBS服务端错误码为SignatureDoesNotMatch，请检查AK/SK是否有误。

17.9 报错 NetworkOnMainThreadException

NetworkOnMainThreadException 一般是因为网络请求在MainThread类中产生的异常。建议和网络请求有关比较耗时的操作，放到一个子线程里，然后用Handler消息机制与主线程通信。

18 常见问题

18.1 SDK 是否支持批量上传、下载或复制对象?

不支持。

目前SDK暂未提供此类接口，您需要自己封装批量上传、下载或复制对象的业务代码。步骤如下：

步骤1 调用**listObjects**列举所有待上传、下载或复制的对象，具体代码示例请参见**列举对象**。

步骤2 对列举出的对象调用单个对象的上传（上传对象）、下载（下载对象）或复制（复制对象）接口。

----结束

以批量上传对象为例，示例代码如下：

```
// 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_ID。
// 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/intl/zh-cn/usermanual-ca/ca_01_0003.html
String ak = System.getenv("ACCESS_KEY_ID");
String sk = System.getenv("SECRET_ACCESS_KEY_ID");
String endPoint = "https://your-endpoint";
// 定义桶内对象的前缀
final String objectPre = "object/";
// 待上传的文件夹，文件夹路径示例：/storage/emulated/0/Pictures，请保证文件夹下面待上传的文件是符合预期的。
// 获取路径方式示例：final String localDirPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "/Picture";
final String localDirPath = Environment.getExternalStorageDirectory().getAbsolutePath() + "localDirPath";

final List<File> list = new ArrayList<>();

public void uploadFiles() {
// 遍历待上传的文件夹，获取所有待上传对象
    File file = new File(localDirPath);
    listFiles(file);

// 创建ObsClient实例
    final ObsClient obsClient = new ObsClient(ak, sk, endPoint);

// 初始化线程池
```



```
    ExecutorService executorService = Executors.newFixedThreadPool(20);

// 执行并发上传
    for (File f : list) {
        executorService.execute(() -> {
            if (f.isDirectory()) {
                // 如果是空文件夹，则在桶内创建对应的空文件夹对象
                String remoteObjectKey = objectPre + f.getPath().substring(localDirPath.length() + 1) + "/";
                obsClient.putObject(bucketName, remoteObjectKey, new ByteArrayInputStream(new byte[0]));
            } else {
                String remoteObjectKey = objectPre + f.getPath().substring(localDirPath.length() + 1);
                obsClient.putObject(bucketName, remoteObjectKey, new File(f.getPath()));
            }
        });
    }

// 等待上传完成
    executorService.shutdown();
    while (!executorService.isTerminated()) {
        try {
            executorService.awaitTermination(5, TimeUnit.SECONDS);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
// 关闭obsClient
    try {
        obsClient.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

// 扫描文件夹下所有对象
void listFiles(File file) {
    File[] fs = file.listFiles();
    assert fs != null;
    if (fs.length < 1) {
        // 如果是空文件夹也需要上传，将其添加到列表中
        list.add(file);
    } else {
        for (File f : fs) {
            if (f.isDirectory()) {
                listFiles(f);
            }
            if (f.isFile()) {
                // 添加待上传对象到列表中
                list.add(f);
            }
        }
    }
}
```

说明

您可以使用多线程并发执行上传/下载/复制操作，以提高效率。

A API 参考

如果您想要了解OBS Android SDK各API的所有参数及定义，请参考**《对象存储服务 Android SDK API参考》**。