对象存储服务

iOS SDK 开发指南

文档版本 01

发布日期 2024-12-09

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 下载

- OBS iOS SDK的最新版本：**OBS iOS SDK**
- SDK API文档地址: **SDK API 文档**

2 兼容性

- iOS兼容性：推荐使用iOS 8.0-12.3.1版本。
- 开发工具兼容性：推荐使用Xcode 8.0-10.0版本。
- 接口函数：与旧版本（V2.1.4）不再兼容。

3 快速入门

3.1 使用前需知

- 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 iOS SDK接口调用的通用方式。

3.2 OBS 服务环境搭建

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**。
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**管理控制台**。
2. 单击页面右上角的“费用和成本”进入费用中心页面。
3. 选择“资金管理 > 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. 在服务列表中选择“对象存储服务 OBS”，开通并进入OBS管理控制台。

步骤3 创建访问密钥

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

- AK：Access Key ID，接入键标识，用户在对象存储服务系统中的接入键标识，一个接入键标识唯一对应一个用户，一个用户可以同时拥有多个接入键标识。对象存储服务系统通过接入键标识识别访问系统的用户。
- SK：Secret Access Key，安全接入键，用户在对象存储服务系统中的安全接入键，是用户访问对象存储服务系统的密钥，用户根据安全接入键和请求头域生成鉴权信息。安全接入键和接入键标识一一对应。

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

- 永久访问密钥：
  a. 登录**管理控制台**。
  b. 单击页面右上角的用户名，并选择“我的凭证”。
  c. 在“我的凭证”页面，单击左侧导航栏的“访问密钥”。
  d. 在“访问密钥”页面，单击“新增访问密钥”。

说明

每个用户最多可创建两个有效的访问密钥。

  e. 在弹出的“新增访问密钥”对话框中，输入描述内容（建议），单击“确定”。

  f. （可选）在弹出的“身份验证”对话框中，选择合适的验证方式进行验证，单击“确定”。

g. 在弹出的“创建成功”提示框中，单击“立即下载”后，密钥会直接保存到浏览器默认的下载文件夹中。

h. 打开下载下来的“credentials.csv”文件即可获取到访问密钥（AK和SK）。

说明

- 在密钥文件中，Access Key ID列对应的值即AK，Secret Access Key列对应的值即SK。
- 为防止访问密钥泄露，建议您将其保存到安全的位置。如果用户在此提示框中单击“取消”，则不会下载密钥，后续也将无法重新下载。如果需要使用访问密钥，可以重新创建新的访问密钥。

- 临时访问密钥：

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

  获取临时访问密钥的接口请参考**获取临时AK/SK和securitytoken**。

须知

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

----结束

3.3 开发环境准备

从**Xcode官网**下载并安装最新版Xcode。

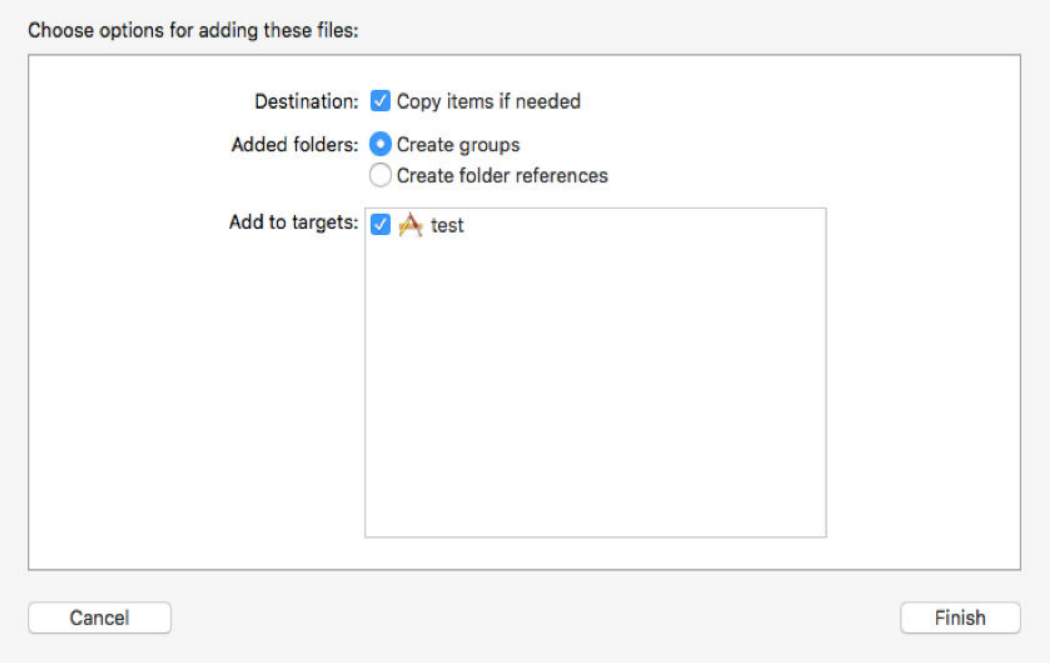
3.4 安装 SDK

iOS 工程配置

步骤1 **下载**OBS iOS SDK软件包。

步骤2 使用Xcode新建工程。

步骤3 将OBS.framework添加到工程中，勾选“Copy items if needed”。

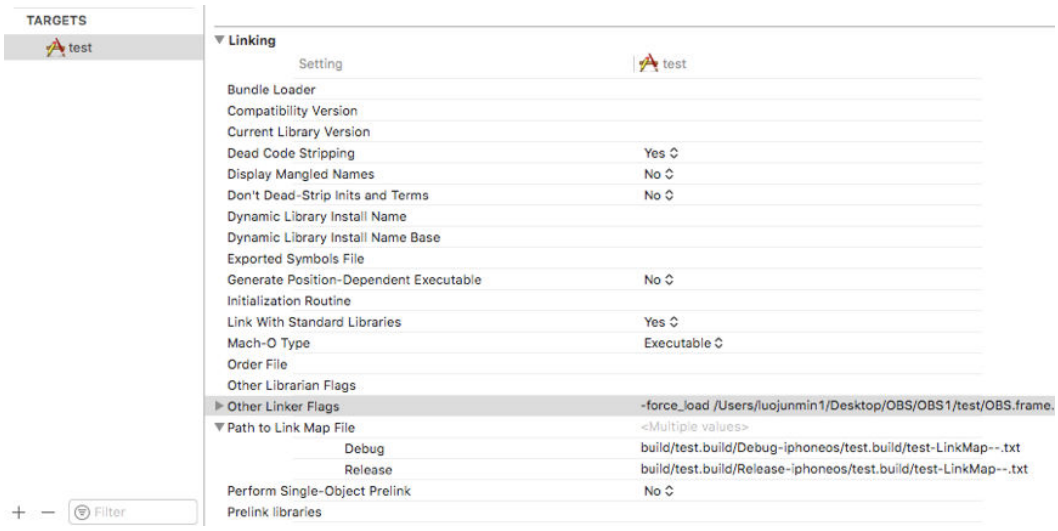


步骤4 在“TARGETS > Build Settings > Linking > Other Linker Flags”添加如下标识：

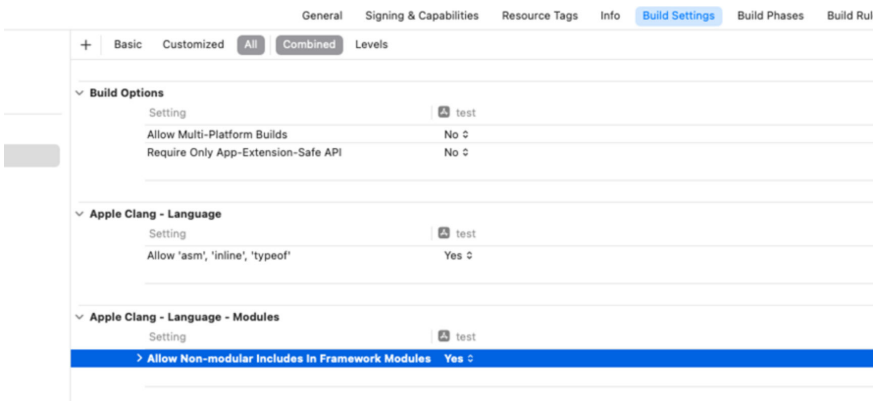
“-force_load

$(SRCROOT)/XXX/OBS.framework/OBS”

其中“*XXX*”代表OBS.framework在工程文件夹下的位置。



步骤5 将“TARGETS > Build Settings > Apple Clang – language –Modules > Allow Non-modular includes in Framework Modules”设置成“Yes”。



步骤6 在需要用到OBS服务的头文件中导入OBS软件包。

```
#import <OBS/OBS.h>
```

步骤7 运行**Command+B**命令确认OBS.framework编译通过。

----结束

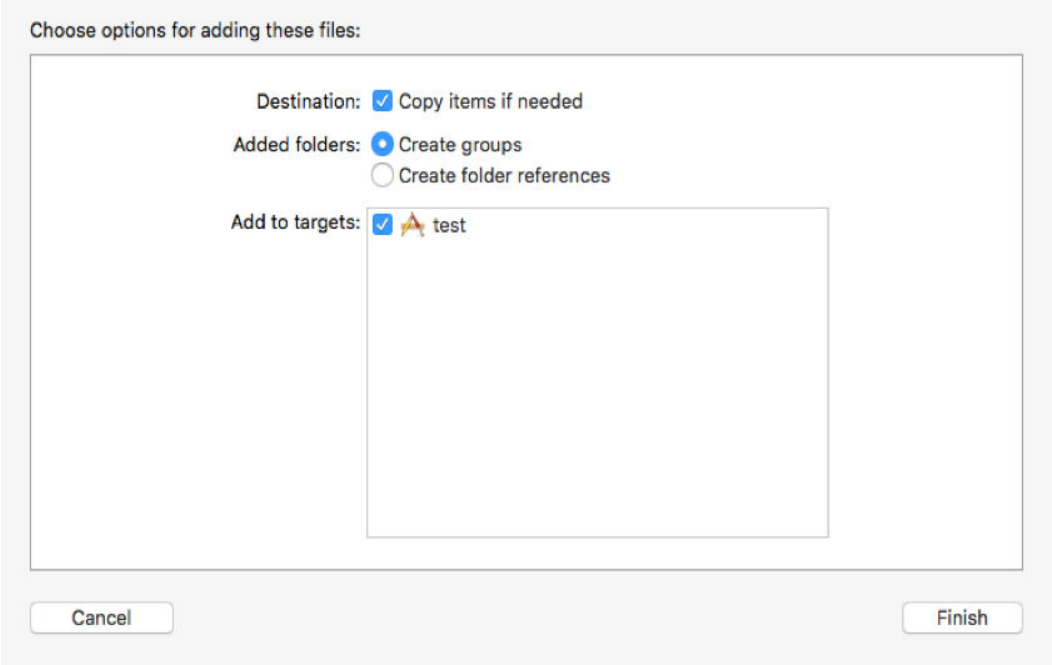
常见问题

Mac OSX 工程配置

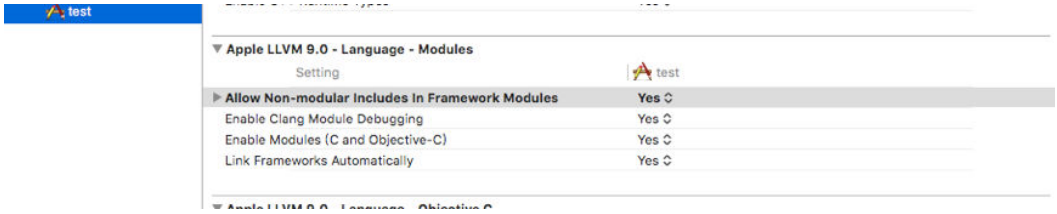
步骤1 **下载**OBS iOS SDK软件包。

步骤2 使用Xcode新建工程。

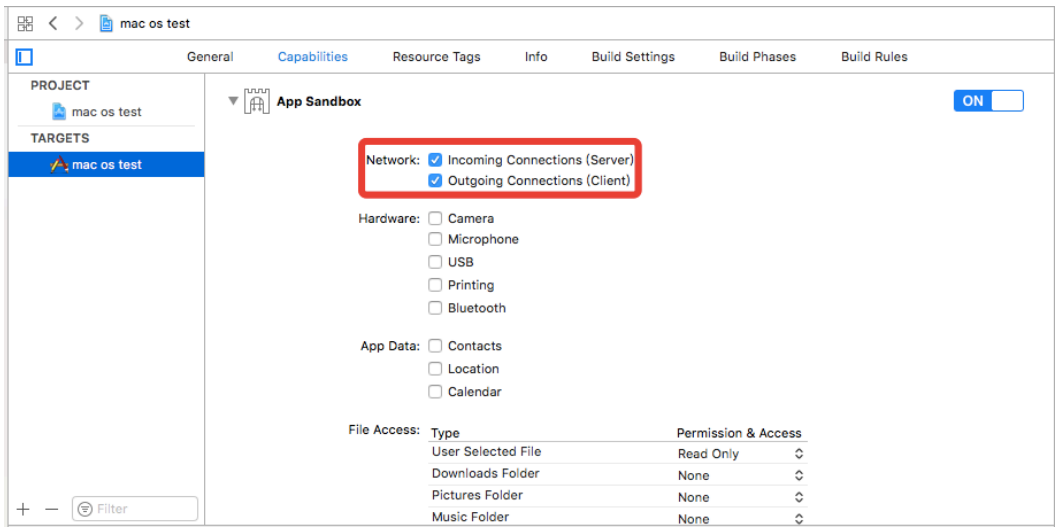
步骤3 将OBS.framework添加到工程中，勾选“Copy items if needed”。



步骤4 在“TARGETS > Build Settings > Apple LLVM9.0 – language –Modules > Allow Non-modular includes in Framework Modules”设置成“Yes”。



步骤5 设置网络连接。



步骤6 在需要用到OBS服务的头文件中导入OBS软件包。

```
#import <OBS/OBS.h>
```

步骤7 运行**Command+B**命令确认OBS.framework编译通过。

----结束

3.5 获取服务地址

- 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需要传入协议，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”两种形式。

3.6 初始化 OBS 客户端

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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:@"https://your-endpoint" credentialProvider:credentialProvider];

// 初始化client
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

说明

更多OBS客户端初始化的内容请参考“初始化”章节。

3.7 创建桶

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：

```
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc] initWithBucketName:@"bucketname"];
[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(response.location);
}];
```

说明

- 桶的名字是全局唯一的，所以您需要确保不与已有的桶名称重复。
- 桶命名规则如下：
  - 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。
  - 禁止使用IP地址。
  - 禁止以“-”或“.”开头及结尾。
  - 禁止两个“.”相邻（如：“my..bucket”）。
  - 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。
- 更多创建桶的信息，请参见**管理桶**。

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

以下代码展示如何上传对象至OBS：

```
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];
// 上传进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};
// 上传文件
[ client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

说明

更多上传对象的信息，请参见**上传对象**。

3.9 下载对象

以下代码展示如何获取对象的内容：

```
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
OBSGetObjectToFileRequest *request = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];
// 下载进度
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

说明

- 调用getObject返回一个OBSObject实例，该实例包含对象内容及其属性。
- 更多下载对象的信息，请参见**下载对象**。

3.10 列举对象

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：

```
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
request.maxKeys = [NSNumber numberWithInt:10];
request.origin = @"www.example1.com";
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    NSLog(@"%d",response.contentsList.count);
}];
```

说明

- 调用listObjects返回OBSListObjectsResponse实例，该实例包含此次listObject请求的返回结果。
- 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

以下代码展示如何删除指定的对象：

```
OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname"
objectKey:@"objectname"];
[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

3.12 OBS 客户端通用示例

调用OBSClient类的相关接口时，没有错误产生则说明调用成功；当有错误产生时，则说明操作失败。以下代码展示了使用OBS客户端的通用方式：

```
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

// 初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
```

```
OBSClient *client = [[OBSClient alloc] initWithConfiguration:conf];


// 创建列举对象请求
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

// 列举对象
OBSBFTask *task = [client listBuckets:request completionHandler:^(OBSListBucketsResponse *response,
NSError *error) {
    if(error){
        // 处理错误

    }else{
        // 获取结果
        for(OBSBucket *bucket in response.bucketsList){
            NSLog(@"bucketname=%@",bucket.name);
        }
    }
}];
[task waitUntilFinished];
```

须知

- 在调用OBSClient类的相关接口时，如果将Client声明为局部变量，则应该通过对请求任务执行waitUntilFinished操作以保证Client在请求任务执行期间始终有效，否则可能造成网络请求失败，程序崩溃的问题。

4 初始化

4.1 配置密钥

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**

4.2 创建 OBS 客户端

OBS客户端（OBSClient）是访问OBS服务的iOS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 iOS 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OBSServiceConfiguration的默认配置项。

永久访问密钥（AK/SK）创建OBS客户端代码如下:

```
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

// 初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

临时访问密钥（AK/SK和SecurityToken）创建OBS客户端代码如下：

```
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];
// securityToken
char* securityToken_env = getenv("SecurityToken");
NSString *SecurityToken = [NSString stringWithUTF8String:securityToken_env];
credentialProvider.securityToken = SecurityToken;

// 初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

说明

当前，对于断点续传任务，当有多个上传任务需并发执行时，需要对每个上传任务初始化一个单独的客户端对请求进行处理。

以自定义域名访问方式创建OBS客户端代码如下：

```
NSString *endPoint = @"your-custom-domain";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

// 初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];
// 设定以自定义域名访问OBS服务
conf.defaultDomainMode = OBSDomainModeCustom;

// 初始化client
OBSClient *client  = [[OBSClient alloc] initWithConfiguration:conf];
```

通过设置OBSServiceConfiguration.defaultDomainMode参数为OBSDomainModeCustom可以指定以自定义域名方式访问OBS服务，此时endPoint参数应设置为指定的自定义域名。

自定义域名访问介绍与配置

当以自定义域名访问OBS桶时，需要先将该自定义域名同对应OBS桶访问域名进行绑定，相关配置请参见**自定义域名绑定简介**，**自定义域名绑定配置**。

注意

当在自定义域名上配置了CDN加速服务，即自定义域名为CDN服务的加速域名时，需要额外对CDN服务进行配置，以保证可以正常使用自定义域名访问OBS服务。

以华为云CDN服务为例，相关配置如下所示：

步骤1 登录华为云CDN服务，从CDN服务左侧列表中选择域名管理项，在该项中可以查看到所有配置的CDN服务域名信息。

步骤2 配置源站。单击要使用的自定义域名项，进入域名配置界面，编辑源站配置，选择主源站类型为源站域名类型，对应源站为要访问的OBS桶域名。

步骤3 配置回源HOST。回源HOST必须指定为加速域名即访问OBS服务时访问的自定义域名，否则可能会出现回源鉴权失败的问题。

----结束

4.3 配置 OBS 客户端

您可通过OBSServiceConfiguration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OBSServiceConfiguration可以设置的参数见下表：

表 4-1 OBS 网络请求配置表

参数	描述	建议值
OBSServiceConfiguration.credentialProvider	用户凭证，参见**表2 OBS服务身份验证配置表**。	N/A
OBSServiceConfiguration.proxyConfig	代理配置，默认为空，参见**表3 代理服务配置表**。	N/A
OBSServiceConfiguration.trustUnsafeCert	是否信任不安全证书，默认为“NO”。	默认
OBSServiceConfiguration.maxConcurrentCommandRequestCount	允许的最大的命令请求并发数，默认为3。	默认
OBSServiceConfiguration.maxConcurrentUploadRequestCount	允许的最大的上传请求并发数，默认为3。	默认
OBSServiceConfiguration.maxConcurrentDownloadRequestCount	允许的最大的下载请求并发数，默认为3。	默认
OBSServiceConfiguration.defaultDomainMode	指定域名访问模式的参数，可设置为OBSDomainModeCustom以使用自定义域名, 默认为非自定义域名访问模式。	默认
OBSServiceConfiguration.commandSessionConfiguration.HTTPMaximumConnectionsPerHost	允许打开的最大的命令请求连接数, ios系统中默认为4。	N/A
OBSServiceConfiguration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost	允许打开的最大的上传请求连接数, ios系统中默认为4。	N/A
OBSServiceConfiguration.downloadSessionConfiguration.HTTPMaximumConnectionsPerHost	允许打开的最大的下载请求连接数, ios系统中默认为4。	N/A
OBSServiceConfiguration.backgroundUploadSessionConfiguration.HTTPMaximumConnectionsPerHost	允许打开的最大的后台上传请求连接数, ios系统中默认为4。	N/A
OBSServiceConfiguration.backgroundDownloadSessionConfiguration.HTTPMaximumConnectionsPerHost	允许打开的最大的后台下载请求连接数, ios系统中默认为4。	N/A
OBSServiceConfiguration.commandSessionConfiguration.timeoutIntervalForRequest	配置命令请求的超时时间；（单位秒）	60
OBSServiceConfiguration.uploadSessionConfiguration.timeoutIntervalForRequest	配置上传相关请求的超时时间；（单位秒）	60

参数	描述	建议值
OBSServiceConfiguration.downloadSessionConfiguration.timeoutIntervalForRequest	配置下载相关请求的超时时间；（单位秒）	60

说明

建议值为N/A的表示需要根据实际情况进行设置。出于安全性考虑，endpoint建议使用https协议。

OBSStaticCredentialProvider可以设置的参数见下表：

表 4-2 OBS 服务身份验证配置表

参数	描述	方法
accessKey	用户的 Access Key。	credentialProvider.Access_Key = Access_Key
secretKey	用户的Secret Key。	credentialProvider.Secret_Key = Secret_Key
securityToken	临时Token	credentialProvider.securityToken = token

说明

credentialProvider是OBSStaticCredentialProvider的实例对象。

securityToken获取方式参见**OBS服务环境搭建**。

OBSHTTPProxyConfiguration可以设置的参数见下表：

表 4-3 代理服务配置表

参数	描述	方法
proxyType	网络访问的类型（枚举类型）。	只允许HTTP： proxyConfig.proxyType=OBSHTTPProxyTypeHTTP 只允许HTTPS： proxyConfig.proxyType=OBSHTTPProxyTypeHTTPS 允许HTTP和HTTPS： proxyConfig.proxyType=OBSHTTPProxyTypeHTTPAndHTTPS
proxyHost	代理服务器的主机地址。	proxyConfig.proxyHost = @"host"

参数	描述	方法
proxyPort	代理服务器的端口号。	proxyConfig.proxyPort = @"port"
username	连接代理服务器时使用的用户名。	proxyConfig.username = @"username"
password	连接代理服务器时使用的用户密码。	proxyConfig.password = @"password"

说明

proxyConfig是OBSHTTPProxyConfiguration实例对象。

5 管理桶

5.1 创建桶

您可以通过createBucket创建桶。

简单创建

以下代码展示如何新建一个桶：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//生成创建桶request
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc] initWithBucketName:@"bucketname"];
//创建桶
[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(@"%@",response.location);
 }];
```


说明

- 桶的名字是全局唯一的，所以您需要确保不与已有的桶名称重复。
- 桶命名规则如下：
  - 3～63个字符，数字或字母开头，支持小写字母、数字、“-”、“.”。
  - 禁止使用IP地址。
  - 禁止以“-”或“.”开头及结尾。
  - 禁止两个“.”相邻（如：“my..bucket”）。
  - 禁止“.”和“-”相邻（如：“my-.bucket”和“my.-bucket”）。
- 同一用户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**默认是公共读写，存储类型默认是标准类型，区域位置是全局域名所在的默认区域。

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。

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//创建桶
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc]
initWithBucketName:@"bucketname"];
// 设置权限为公共读写
request.bucketACLPolicy = OBSACLPolicyPublicReadWrite;
// 存储模式设置为标准存储
request.defaultStorageClass = OBSStorageClassStandard;
// 设置桶区域配置
request.configuration = [[OBSBucketConfiguration alloc] initWithLocationConstraint:@"bucketlocation"];
```

```
[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(@"%@",response.location);
}];
```

5.2 列举桶

您可以通过listBuckets来列举桶。以下代码展示如何获取桶列表：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举桶
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    for(OBSBucket *bucket in response.bucketsList){
        NSLog(@"bucketname=%@",bucket.name);
    }
}];
```

说明

获取到的桶列表将按照桶名字典顺序排列。

5.3 删除桶

您可以通过deleteBucket来删除桶。以下代码展示如何删除桶：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
```



```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//删除桶
OBSDeleteBucketRequest *deleteRequest = [[OBSDeleteBucketRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucket:deleteRequest completionHandler:^(OBSDeleteBucketResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

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 获取桶元数据

您可以通过getBucketMetaData获取桶元数据。以下代码展示如何获取桶元数据：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//获取桶元数据
OBSGetBucketMetaDataRequest *request = [[OBSGetBucketMetaDataRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketMetaData:request completionHandler:^(OBSGetBucketMetaDataResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```

5.5 管理桶访问权限

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OBSSetBucketACLWithCannedACLRequest指定预定义访问策略。

3. 调用OBSSetBucketACLWithPolicyRequest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 iOS SDK对应枚举值
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。	OBSACLRead
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。	OBSACLWrite
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。	OBSACLRead_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。	OBSACLWrite_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	OBSACLFull_Control

OBS预定义的访问策略包含五类，见下表：

权限	描述	OBS iOS SDK对应枚举值
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。	OBSACLPolicyPrivate
公共读	桶或对象的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读的权限。	OBSACLPolicyPublicRead
公共读写	桶或对象的所有者拥有完全控制的权限，其他所有用户包括匿名用户拥有读和写的权限。	OBSACLPolicyPublicReadWrite

权限	描述	OBS iOS SDK对应枚举值
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。	OBSACLPolicyPublicReadDelivered
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。	OBSACLPolicyPublicReadWriteDelivered

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//创建桶
OBSCreateBucketRequest *request = [[OBSCreateBucketRequest alloc] initWithBucketName:@"bucketname"];
// 设置权限为公有读写
request.bucketACLPolicy = OBSACLPolicyPublicReadWrite;

[client createBucket:request completionHandler:^(OBSCreateBucketResponse *response, NSError *error) {
    NSLog(@"%@",response.location);
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//设置桶的预定义访问策略  公共读私有写
OBSSetBucketACLWithCannedACLRequest *request = [[OBSSetBucketACLWithCannedACLRequest alloc]initWithBucketName:@"bucketname" cannedACL:OBSACLPolicyPublicRead];
[client setBucketACL:request completionHandler:^(OBSSetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 初始化拥有者实例
OBSUser *owner = [[OBSUser alloc] initWithID:@"ownerID"];
// 设置授权用户
OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"granteeID"];
// 授权用户设置完全控制权限
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

// 生成策略对象
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];

// 直接设置桶访问策略
OBSSetBucketACLWithPolicyRequest *request = [[OBSSetBucketACLWithPolicyRequest alloc]initWithBucketName:@"bucketname" accessControlPolicy:policy];
```



```
[client setBucketACL:request completionHandler:^(OBSSetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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getBucketACL获取桶的访问权限。以下代码展示如何获取桶访问权限：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶策略
OBSGetBucketACLRequest *request = [[OBSGetBucketACLRequest alloc] initWithBucketName:@"bucketname"];
[client getBucketACL:request completionHandler:^(OBSGetBucketACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

5.6 获取桶区域位置

您可以通过getBucketLocation获取桶的区域位置。以下代码展示如何获取桶区域位置：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
```

```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶区域
OBSGetBucketLocationRequest *request = [[OBSGetBucketLocationRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketLocation:request completionHandler:^(OBSGetBucketLocationResponse *response, NSError
*error){
    NSLog(response.configuration.locationConstraint);
}];
```

说明

创建桶时可以指定桶的区域位置，请参见**创建桶**。

5.7 管理桶策略

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setBucketPolicy设置桶策略。示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置桶策略
OBSSetBucketPolicyWithStringRequest *request = [[OBSSetBucketPolicyWithStringRequest alloc]
initWithBucketName:@"bucketname" policyString:@"policystring"];
[client setBucketPolicy:request completionHandler:^(OBSSetBucketPolicyResponse *response, NSError *error)
{
    NSLog(@"%@",response);
 }];
```

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getBucketPolicy获取桶策略。示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶策略
OBSGetBucketPolicyRequest *request = [[OBSGetBucketPolicyRequest alloc] initWithBucketName:g_bucketName];
[client getBucketPolicy:request completionHandler:^(OBSGetBucketPolicyResponse *response, NSError *error){
    NSLog(@"%@",response);
 }];
```

删除桶策略

您可以通过deleteBucketPolicy删除桶策略。示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 删除桶策略
OBSDeleteBucketPolicyRequest *request = [[OBSDeleteBucketPolicyRequest alloc] initWithBucketName:g_bucketName];
[client deleteBucketPolicy:request completionHandler:^(OBSDeleteBucketPolicyResponse *response, NSError *error) {
```



```
    NSLog(@"%@",response);
}];
```

5.8 获取桶存量信息

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getBucketStorageInfo获取桶的存量信息。以下代码展示如何获取桶存量信息：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶存量信息
OBSGetBucketStorageInfoRequest *request = [[OBSGetBucketStorageInfoRequest alloc] initWithBucketName:@"bucketname"];

[client getBucketStorageInfo:request completionHandler:^(OBSGetBucketStorageInfoResponse *response, NSError *error) {
    NSLog(@"%@",response.storageInfo);
 }];
```

5.9 桶配额

设置桶配额

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
```



```
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置桶配额
OBSQuota *quota = [[OBSQuota alloc] initWithQuotaNumber:[NSNumber
numberWithLongLong:1024*1024*1024]];

OBSSetBucketQuotaRequest *request = [[OBSSetBucketQuotaRequest alloc]
initWithBucketName:@"bucketname" quota:quota];
[client setBucketQuota:request completionHandler:^(OBSSetBucketQuotaResponse *response, NSError
*error) {
    NSLog(@"%@",response.statusCode);
}];
```

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getBucketQuota获取桶配额。以下代码展示如何获取桶配额：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶配额
OBSGetBucketQuotaRequest *request = [[OBSGetBucketQuotaRequest
alloc]initWithBucketName:@"bucketname"];

[client getBucketQuota:request completionHandler:^(OBSGetBucketQuotaResponse *response, NSError
*error){
    NSLog(@"%@",response.quota);
}];
```

5.10 桶存储类型

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 iOS SDK对应枚举值
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。	OBSStorageClassStandard
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。	OBSStorageClassStandardIA
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。	OBSStorageClassGlacier

更多关于桶存储类型的内容请参考**桶的存储类别**。

设置桶存储类型

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置桶为标准存储类型
OBSStoragePolicy* policy = [[OBSStoragePolicy alloc]initWithStorageClass:OBSStorageClassStandard];

// 设置桶的存储类型
OBSSetBucketStoragePolicyRequest *request = [[OBSSetBucketStoragePolicyRequest alloc]initWithBucketName:@"bucketname" storagePolicy:policy];

[client setBucketStoragePolicy:request completionHandler:^(OBSSetBucketStoragePolicyResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

获取桶存储类型

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶的存储类型
OBSGetBucketStoragePolicyRequest *request = [[OBSGetBucketStoragePolicyRequest alloc]initWithBucketName:@"bucketname"];

[client getBucketStoragePolicy:request completionHandler:^(OBSGetBucketStoragePolicyResponse *response, NSError *error){
    NSLog(@"%@",response.storagePolicy);
}];
```


6 上传对象

6.1 对象上传简介

在OBS中，用户操作的基本数据单元是对象。OBS iOS 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**

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。

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。

6.2 流式上传

流式上传使用OBSPutObjectWithDataRequest作为对象的数据源。您可以通过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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];
```

```
// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式上传字符串
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadData:[@"hello"
dataUsingEncoding:NSUTF8StringEncoding]];

// 上传文件进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

上传网络流

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式上传网络流
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest alloc]
initWithBucketName:@"bucketname" objectKey:@"objectname" uploadDataURL:[NSURL
URLWithString:@"DateURL"]];

// 上传文件进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   NSLog(@"%@",response);
}];
```


上传文件流

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// 文件流上传
NSData *uploadData = [NSData dataWithContentsOfFile:filePath];
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"test/image1" uploadData:uploadData];

// 上传文件进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
   NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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。

6.3 文件上传

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```

```
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
//指定待上传文件名及文件类型
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// 文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucket-ios-test02" objectKey:@"imageWithFile" uploadFilePath:filePath];
// 开启后台上传，当应用退出到后台后，上传任务仍然会进行
request.background = YES;

// 上传进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    // 判断error状态
    if(error){
        // 打印错误信息
        NSLog(@"文件上传失败");
        NSLog(@"%@",error);
    }
    // 上传文件成功，返回200，打印返回响应值
    if([response.statusCode isEqualToString:@"200"]){
        NSLog(@"文件上传成功");
        NSLog(@"%@",response);
        NSLog(@"%@",response.etag);
    }
}];
```

说明

上传的内容大小不能超过5GB。

设置background为YES时，可以开启后台上传。

6.4 创建文件夹

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。

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];
//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
```

```
credentialProvider:credentialProvider];
// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 创建一个空字符串
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"file/" uploadData:[@""
dataUsingEncoding:NSUTF8StringEncoding]];

// 上传文件进度
request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};
//创建一个file文件夹
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   NSLog(@"%@",response);
}];
```

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/。

6.5 设置对象属性

您可以在上传对象时设置对象属性。对象属性包含对象MD5值（用于校验）、对象存储类型、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
contentMD5	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
storageClass	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。	标准存储
metaDataDict	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
contentType	上传对象时指定的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。	二进制流
customContentType	上传对象时指定的MIME类型，定义对象的类型及网页编码。区别于contentType参数，customContentType参数允许传入任意字符以指定上传对象的MIME类型。	无

设置对象 MD5 值

您可以通过contentMD5来设置对象MD5值。以下代码展示如何设置对象MD5值：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
//文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

//设置对象MD5
request.contentMD5 = @"your md5 which should be encoded by base64"

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

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。

设置对象存储类型

您可以通过storageClass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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];
```



```
// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
//文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// 设置对象存储类型
request.storageClass = OBSStorageClassStandard;

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

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，参见**下载归档存储对象**。

设置对象自定义元数据

您可以通过metaDataDict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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式上传字符串
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadData:[@""
dataUsingEncoding:NSUTF8StringEncoding]];

// 设置对象元数据
request.metaDataDict = @{@"meta1":@"value1",@"meta2":@"value2"};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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“meta1”，值为“value1”的元数据和一个名称为“meta2”，值为“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OBSGetObjectMetaDataRequest获取，参见**获取对象元数据**。
- 使用getObject下载对象时，对象的自定义元数据也会同时下载。

设置对象上传类型

您可以通过contentType来设置对象上传类型。以下代码展示如何通过contentType参数设置对象上传类型：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"mp4"];
//文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// 设置对象上传类型
request.contentType = OBSContentTypeMP4;

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

类型	OBS IOS SDK对应枚举值
video/mp4	OBSContentTypeMP4
text/html	OBSContentTypeHTML
image/png	OBSContentTypePNG
image/jpeg	OBSContentTypeJPEG
image/gif	OBSContentTypeGIF
application/pdf	OBSContentTypePDF
audio/mp3	OBSContentTypeMP3

类型	OBS IOS SDK对应枚举值
audio/wav	OBSContentTypeWAV
binary/octet-stream	OBSContentTypeBinary
video/quicktime	OBSContentTypeMOV
application/vnd.apple.mpegurl	OBSContentTypeM3U8

您还可以通过customContentType字段来设置对象上传类型，以下代码展示如何通过customContentType参数设置对象上传类型：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"mp4"];
//文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// 设置对象上传类型
request.customContentType = @"video/mp4";

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

说明

- 当前contentType参数支持指定上表中列出的11种对象类型。如果不设置，默认采用binary/octet-stream类型。
- 区别于contentType参数，customContentType参数支持以字符串形式指定任意对象上传类型。如果不设置，以contentType对应类型为对象最终上传类型。
- 当同时指定了customContentType参数及contentType参数时，对象最终上传类型为customContentType指定的上传类型。

6.6 分段上传

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（initiateMultipartUpload）。

步骤2 逐个或并行上传段（uploadPart）。

步骤3 合并段（completeMultipartUpload）或取消分段上传任务(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initiateMultipartUpload初始化一个分段上传任务：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 初始化多段上传任务
OBSInitiateMultipartUploadRequest *request = [[OBSInitiateMultipartUploadRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

[client initiateMultipartUpload:request completionHandler:^(OBSInitiateMultipartUploadResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

说明

- response.uploadID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。
- 在OBSInitiateMultipartUploadRequest中，您可以设置对象存储类型、对象自定义元数据等对象属性。同时您还可以通过customContentType参数来设置对象上传类型。
- 设置background为YES时，可以开启后台上传。

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uploadPart上传段：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// 第一段
OBSUploadPartWithFileRequest* fileRequest_first =
[[OBSUploadPartWithFileRequest alloc]initWithBucketName:@"bucket-ios-test03"
                                              objectkey:@"MultiPart"
                                             partNumber:[NSNumber numberWithInt:1]
                                               uploadID:@"uploadID"
                                         uploadFilePath:filePath];
// 开启后台上传，当应用退出到后台后，上传任务仍然会进行
fileRequest_first.background = YES;

[client uploadPart:fileRequest_first completionHandler:^(OBSUploadPartResponse *response, NSError *error)
{
    NSLog(@"第一段");
}];

// 第二段
OBSUploadPartWithFileRequest* fileRequest_sec =
[[OBSUploadPartWithFileRequest alloc]initWithBucketName:@"bucket-ios-test03"
                                              objectkey:@"MultiPart"
                                             partNumber:[NSNumber numberWithInt:2]
                                               uploadID:@"uploadID"
                                         uploadFilePath:filePath];

[client uploadPart:fileRequest_sec completionHandler:^(OBSUploadPartResponse *response, NSError *error)
{
    NSLog(@"第二段");
}];
```


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。
- 分段号的范围是1~10000。如果超出这个范围，OBS将返回400 Bad Request错误。
- OBS 3.0的桶支持最小段的大小为100KB，OBS 2.0的桶支持最小段的大小为5MB。请在OBS 3.0的桶上执行分段上传操作。

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completeMultipartUpload合并段：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举段
__block NSMutableArray *partsList;
OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:@"uploadID"];
OBSBFTask *listTask = [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
    partsList = [response.partsList mutableCopy];

        // 合并段
    OBSCompleteMultipartUploadRequest* comRequest = [[OBSCompleteMultipartUploadRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:@"uploadID"];
    comRequest.partsList = partsList;
   [client completeMultipartUpload:comRequest completionHandler:^(OBSCompleteMultipartUploadResponse *response, NSError *error) {
            NSLog(@"%@",response);
    }];

}];
```


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abortMultipartUpload取消分段上传任务：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 取消分段上传任务
OBSAbortMultipartUploadRequest *abortRequest = [[OBSAbortMultipartUploadRequest alloc] initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:@"uploadID"];

[client abortMultipartUpload:abortRequest completionHandler:^(OBSAbortMultipartUploadResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

列举已上传的段

您可使用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 iOS SDK对应方法
bucketName	分段上传任务所属的桶名。	request.bucketName
objectKey	分段上传任务所属的对象名。	request.objectKey
uploadID	分段上传任务全局唯一标识，从initiateMultipartUpload返回的结果获取。	request.uploadID

参数	作用	OBS iOS 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。	request.maxParts
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。	request.partNumberMarker

- 简单列举

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];


OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:uploadID];
[client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

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response.isTruncated为YES表明本次没有返回全部段，并可通过response.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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];


__block OBSListPartsResponse *result;

OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadID:uploadID];

// 列举所有上传段
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
        result = response;
        NSLog(@"%@",result);
        listRequest.partNumberMarker = result.nextPartNumberMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);

} while (result.isTruncated);
```

- 分页列举所有段

上面的获取所有已上传段（每页1000个段）是分页的一种特殊情况。如果需要指定每页段的数量，请参考以下代码：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
NSString *uploadID = @"upload id from OBSInitateMultipartUpload";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

 __block OBSListPartsResponse *result;
```



```
OBSListPartsRequest* listRequest = [[OBSListPartsRequest alloc]initWithBucketName:@"bucketname"
objectKey:@"objectname" uploadID:uploadID];
listRequest.maxUploads = [NSNumber numberWithInt:100];
// 列举所有上传段
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listParts:listRequest completionHandler:^(OBSListPartsResponse *response, NSError *error) {
        result = response;
        NSLog(@"%@",result);
        listRequest.partNumberMarker = result.nextPartNumberMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);

} while (result.isTruncated);
```

列举分段上传任务

您可以通过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 iOS SDK对应方法
bucketName	桶名。	request.bucketName = @"bucketname"
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。	request.delimiter = @"delimiter"
prefix	限定返回的分段上传任务中的对象名必须带有prefix前缀。	request.prefix = @"prefix"
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。	request.maxUploads = [NSNumber numberWithInt:Int]
keyMarker	表示列举时返回指定的keyMarker之后的分段上传任务。	request.keyMarker = @"keymarker"
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。	request.uploadIDMarker = @"ifmarker"

- 简单列举分段上传任务

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举分段上传任务
OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest alloc]initWithBucketName:@"bucketname"];

[client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

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response.isTruncated为YES表明本次没有返回全部结果，并可通过response.nextKeyMarker和response.nextUploadIdMarker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。

- 列举全部分段上传任务

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
__block OBSListMultipartUploadsResponse *result;
OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest alloc]initWithBucketName:@"bucket-ios-test03"];

// 列举全部分段上传任务
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response, NSError *error) {
        result = response;
        NSLog(@"%@",result);
        request.keyMarker = result.nextKeyMarker;
```



```
        request.uploadIDMarker = result.nextUploadIDMarker;
        dispatch_semaphore_signal(sema);
        }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} while (result.isTruncated);
```

- 分页列举全部分段上传任务

上面的获取所有分段上传任务（每页1000个任务）是分页的一种特殊情况。如果需要指定每页任务的数量，请参考以下代码：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];


__block OBSListMultipartUploadsResponse *result;


OBSListMultipartUploadsRequest *request = [[OBSListMultipartUploadsRequest alloc]initWithBucketName:@"bucketname"];

// 每页100个分段上传任务
request.maxUploads = [NSNumber numberWithInt:100];

// 列举全部分段上传任务
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listMultipartUploads:request completionHandler:^(OBSListMultipartUploadsResponse *response, NSError *error) {
        result = response;
        NSLog(@"%@",result);
        request.keyMarker = result.nextKeyMarker;
        request.uploadIDMarker = result.nextUploadIDMarker;
        dispatch_semaphore_signal(sema);

    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);

} while (result.isTruncated);
```

6.7 设置对象生命周期

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
// 文件上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];

// 设置30天后过期
request.expires = [NSNumber numberWithInteger:30];

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
   NSLog(@"%@",response.etag);
}];
```

说明

- 上述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。
- 上述方式设置的对象过期时间，其优先级高于桶生命周期规则。

6.8 追加上传

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appendObject进行追加上传。示例代码如下：

```
   NSString *filePath = [[NSBundle mainBundle] pathForResource:@"FileName" ofType:@"FileSuffix"];
   NSFileManager *manager = [NSFileManager defaultManager];
   NSDictionary *fileDic = [manager attributesOfItemAtPath:filePath error:nil];
   unsigned long long size = [[fileDic objectForKey:NSFileSize] longLongValue];
   int filesize = size;
   //第一次追加上传
   OBSAppendObjectWithFileRequest *request = [[OBSAppendObjectWithFileRequest alloc] initWithBucketName:@"bucketName" objectKey:@"objectname" uploadFilePath:filePath];
   request.position = [NSNumber numberWithFloat:0];

   request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
       NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);

   };

   __block NSString* nextPosition = nil;
   [client appendObject:request completionHandler:^(OBSAppendObjectResponse *response, NSError *error) {
       NSLog(@"%@",response);
       //下次上传位置
       NSDictionary *temp = [response headers];
       nextPosition = [temp valueForKey:@"x-obs-next-append-position"];
       NSLog(@"nextPosition:%@", nextPosition);
   }];

   //第二次追加上传
```



```
    request = [[OBSAppendObjectWithFileRequest alloc] initWithBucketName:@"bucketName"
objectKey:@"objectname" uploadFilePath:filePath];

    int nextPositionInt = [nextPosition intValue];
    request.position = [NSNumber numberWithInt:nextPositionInt];
    request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t
totalBytesExpectedToSend) {
        NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);

    };
    [client appendObject:request completionHandler:^(OBSAppendObjectResponse *response, NSError
*error) {
        NSLog(@"%@",response);
        //下次上传位置
        NSDictionary *temp = [response headers];
        nextPosition = [temp valueForKey:@"x-obs-next-append-position"];
        NSLog(@"nextPosition:%@", nextPosition);
    }];
```

说明

- putObject上传的对象可覆盖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是当次追加数据内容的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
  ```
  NSDictionary *temp = [response headers];NSString* nextPosition = [temp valueForKey:@"x-obs-next-append-position"];
  ```
  这种方式获取下次追加上传的位置；或者通过getObjectMetadata接口获取下次追加上传的位置。

6.9 分段复制

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copyPart来复制段。以下代码展示了如何使用分段复制对象：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 拷贝段
OBSCopyPartRequest* copyRequest =
[[OBSCopyPartRequest alloc]initWithSrcBucketName:@"buckeName"
```



```
                                    srcObjectKey:@"MultiPart"
                                uploadBucketName:@"bucketName"
                                 uploadObjectKey:@"MultiPart"
                                uploadPartNumber:[NSNumber numberWithInt:3]
                                        uploadID:@"uploadID"];

[client copyPart:copyRequest completionHandler:^(OBSCopyPartResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

6.10 断点续传上传

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 iOS SDK对应方法
bucketName	桶名，必选参数。	request.bucketName
objectKey	对象名，必选参数。	request.objectKey
objectACLPolicy	对象访问策略。	request.objectACLPolicy
storageClass	对象存储类型。	request.storageClass
metaDataDict	对象元数据。	request.metaDataDict
websiteRedirectLocation	网址重定向位置。	request.websiteRedirectLocation
encryption	加密方式。	request.encryption
enableCheckpoint	是否开启断点续传模式，默认为NO，表示不开启。	request.enableCheckpoint
enableMD5Check	是否开启MD5校验，默认为NO，表示不开启。	request.enableMD5Check
checkpointFilePath	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。文件名后缀可指定为obsuploadcheckpoint。	request.checkpointFilePath
partSize	分段大小，单位字节，取值范围是100KB~5GB，默认为5MB。	request.partSize

以下代码展示了如何使用断点续传上传接口上传文件：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 分段上传的最大并发数
client.configuration.maxConcurrentUploadRequestCount = 5;
// 分段上传请求的最大连接数
client.configuration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost = 10;
NSString *filePath = [[NSBundle mainBundle]pathForResource:@"fileName" ofType:@"Type"];
OBSUploadFileRequest *request = [[OBSUploadFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:filePath];
// 分段大小为5MB
request.partSize = [NSNumber numberWithInteger: 5 * 1024*1024];
// 开启断点续传模式
request.enableCheckpoint = YES;
// 指定checkpoint文件路径
request.checkpointFilePath = @"Your CheckPoint File";

// 上传文件
request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

OBSBFTask  *task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response, NSError *error) {
     NSLog(@"%@",response);
    if(error){
        // 再次上传
    }
}];
```

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NO，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**参数仅在enableCheckpoint参数为YES时有效。
- 当前，当有多个上传任务需要并发执行时，需为每个上传任务初始化一个client及request进行处理。

6.11 暂停、继续、取消断点续传上传

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 iOS SDK对应方法
bucketName	桶名，必选参数。	request.bucketName
objectKey	对象名，必选参数。	request.objectKey
objectACLPolicy	对象访问策略。	request.objectACLPolicy
storageClass	对象存储类型。	request.storageClass
metaDataDict	对象元数据。	request.metaDataDict
websiteRedirectLocation	网址重定向位置。	request.websiteRedirectLocation
encryption	加密方式。	request.encryption
enableCheckpoint	是否开启断点续传模式，默认为NO，表示不开启。	request.enableCheckpoint
enableMD5Check	是否开启MD5校验，默认为NO，表示不开启。	request.enableMD5Check
checkpointFilePath	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。文件名后缀可指定为obsuploadcheckpoint。	request.checkpointFilePath
partSize	分段大小，单位字节，取值范围是100KB~5GB，默认为5MB。	request.partSize
needAbortUploadFileAfterCancel	是否在断点续传上传暂停时取消分段整个上传任务	request.needAbortUploadFileAfterCancel

以下代码展示了如何暂停、继续断点续传上传接口：

```
#import <OBS/OBS.h>
void testPauseAndResumeUploadFile(){
    static OBSClient *client;
    NSString *endPoint = @"your-endpoint";
```

```
    // 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
    char* ak_env = getenv("AccessKeyID");
    char* sk_env = getenv("SecretAccessKey");
    NSString *AK = [NSString stringWithUTF8String:ak_env];
    NSString *SK = [NSString stringWithUTF8String:sk_env];

    // 初始化身份验证
    OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

    //初始化服务配置
    OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

    // 初始化client
    client = [[OBSClient alloc] initWithConfiguration:conf];

    // 分段上传的最大并发数
    client.configuration.maxConcurrentUploadRequestCount = 5;
    // 分段上传请求的最大连接数
    client.configuration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost = 10;
    NSString *filePath = [[NSBundle mainBundle]pathForResource:@"localFile" ofType:@"fileType"];
    OBSUploadFileRequest *request = [[OBSUploadFileRequest alloc]initWithBucketName:@"your-bucket-name" objectKey:@"objectname" uploadFilePath:filePath];
    // 分段大小为5MB
    request.partSize = [NSNumber numberWithInteger: 5 * 1024*1024];
    // 开启断点续传模式
    request.enableCheckpoint = YES;
    // 指定checkpoint文件路径，可以不设置
//    request.checkpointFilePath = @"Your CheckPoint File";

    // 上传文件
    request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
        NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
    };

    OBSBFTask  *task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response, NSError *error) {
        if(error){
            // 暂停时此处会打印暂停导致的报错信息
            NSLog(@"upload file failed:%@", error);
        }
        if(response){
            NSLog(@"upload file response:%@", response);
        }
    }];
    // 上传1秒后暂停
    sleep(1);
    [request cancel];
    // 暂停上传后，可以继续上传
    [task waitUntilFinished];
    task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response, NSError *error) {
        if(error){
            NSLog(@"upload file failed:%@", error);
            // 再次上传
        }
        if(response){
            NSLog(@"upload file response:%@", response);
        }
    }];
    [task waitUntilFinished];
}
```

以下代码展示了如何暂停并取消断点续传上传接口：


```
#import <OBS/OBS.h>
void testPauseAndAbortUploadFile() {
    static OBSClient *client;
    NSString *endPoint = @"your-endpoint";
    // 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
    char* ak_env = getenv("AccessKeyID");
    char* sk_env = getenv("SecretAccessKey");
    NSString *AK = [NSString stringWithUTF8String:ak_env];
    NSString *SK = [NSString stringWithUTF8String:sk_env];

    // 初始化身份验证
    OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

    //初始化服务配置
    OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

    // 初始化client
    client = [[OBSClient alloc] initWithConfiguration:conf];

    // 分段上传的最大并发数
    client.configuration.maxConcurrentUploadRequestCount = 5;
    // 分段上传请求的最大连接数
    client.configuration.uploadSessionConfiguration.HTTPMaximumConnectionsPerHost = 10;
    NSString *filePath = [[NSBundle mainBundle]pathForResource:@"localFile" ofType:@"fileType"];
    OBSUploadFileRequest *request = [[OBSUploadFileRequest alloc]initWithBucketName:@"your-bucket-name" objectKey:@"objectname" uploadFilePath:filePath];
    // 分段大小为5MB
    request.partSize = [NSNumber numberWithInteger: 5 * 1024*1024];
    // 开启断点续传模式
    request.enableCheckpoint = YES;
    // 指定checkpoint文件路径，可以不设置
//    request.checkpointFilePath = @"Your CheckPoint File";

    // 上传文件
    request.uploadProgressBlock =  ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
        NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
    };
    // 暂停后自动取消分段上传任务
    request.needAbortUploadFileAfterCancel = YES;

    OBSBFTask  *task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response, NSError *error) {
        if(error){
            // 暂停时此处会打印暂停导致的报错信息
            NSLog(@"upload file failed:%@", error);
        }
        if(response){
            NSLog(@"upload file response:%@", response);
        }
    }];
    // 上传1秒后暂停，并会自动取消分段上传任务
    sleep(1);
    [request cancel];
    // 暂停上传后，可以重新继续上传，但是由于取消了分段上传任务，进度会清零，从头开始
    [task waitUntilFinished];
    task = [client uploadFile:request completionHandler:^(OBSUploadFileResponse *response, NSError *error) {
        if(error){
            NSLog(@"upload file failed:%@", error);
            // 再次上传
        }
        if(response){
```



```
        NSLog(@"upload file response:%@", response);
    }
  }];
  [task waitUntilFinished];
}
```

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NO，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**参数仅在enableCheckpoint参数为YES时有效。
- 当前，当有多个上传任务需要并发执行时，需为每个上传任务初始化一个client及request进行处理。
- needAbortUploadFileAfterCancel参数默认为NO，如果同时开启了断点续传（**enableCheckpoint**为YES）那么暂停断点续传时不会取消整个分段上传任务，不会丢弃已经上传的段，其他情况暂停时都会取消整个上传段任务。

7 下载对象

7.1 下载对象简介

OBS iOS 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getObject下载对象。

7.2 流式下载

以下代码展示了如何进行流式下载：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式下载
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// 下载进度
```

```
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// 下载的数据
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// 下载结果
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
}
```

7.3 范围下载

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式下载
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// 设置下载范围0-10000
request.range = @"bytes=0-1000";

// 下载进度
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// 下载的数据
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// 下载结果
```



```
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

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。

7.4 限定条件下载

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 iOS SDK对应方法
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。	request.ifModifiedSince
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。	request.ifUnmodifiedSince
ifETagMatch	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。	request.ifETagMatch
ifETagNoneMatch	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。	request.ifETagNoneMatch

说明

- 对象的ETag值是指对象数据的MD5校验值。
- 如果限制条件不符合，则会返回错误preconditionFailed。

以下代码展示了如何进行限定条件下载：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
```

```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式下载
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// 限定条件
request.ifModifiedSince = [[OBSUtils getDateFormatterRFC1123]dateFromString:@"Mon, 18 Dec 2017
03:50:49 GMT"];
// Etag相等
request.ifETagMatch = @"123223";


// 下载进度
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// 下载的数据
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// 下载结果
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

说明

当使用OBSGetObjectToFileRequest时，可以设置background属性为YES进行后台下载。

7.5 重写响应头

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 iOS SDK对应方法
responseContentType	重写HTTP/HTTPS响应中的Content-Type	request.responseContentType
responseContentLanguage	重写HTTP/HTTPS响应中的Content-Language	request.responseContentLanguage
responseExpires	重写HTTP/HTTPS响应中的Expires	request.responseExpires
responseCacheControl	重写HTTP/HTTPS响应中的Cache-Control	request.responseCacheControl
responseContentDisposition	重写HTTP/HTTPS响应中的Content-Disposition	request.responseContentDisposition
responseContentEncoding	重写HTTP/HTTPS响应中的Content-Encoding	request.responseContentEncoding

以下代码展示了如何重写响应头：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 流式下载
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

//重写ContentType
request.responseContentType = @"image/jpeg";


// 下载进度
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};

// 下载的数据
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

// 下载结果
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

7.6 获取自定义元数据

在下载对象成功后response中包含了对象的元数据(mataDataDict)。

7.7 下载归档存储对象

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二类，见下表：

选项	说明	OBS iOS SDK对应值
快速恢复	恢复耗时1~5分钟。	OBSRestoreTierExpedited

选项	说明	OBS iOS SDK对应值
标准恢复	恢复耗时3~5小时。默认值。	OBSRestoreTierStandard

⚠ 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。产生的标准存储类别的对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

您可以通过OBSRestoreObjectRequest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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 恢复对象
OBSRestoreObjectRequest *request = [[OBSRestoreObjectRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" storeDays:[NSNumber
numberWithInt:30]]; //1 to 30
request.restoreTier = OBSRestoreTierExpedited;

OBSBFTask *task = [ self.client restoreObject:request completionHandler:^(OBSRestoreObjectResponse
*response, NSError *error){
    NSLog(@"%@",response);
}];

// 等待对象恢复
sleep(6*60);

// 下载对象
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
OBSGetObjectToFileRequest *request1 = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];
request1.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};

[client getObject:request1 completionHandler:^(OBSGetObjectResponse *response, NSError *error){
```

```
    NSLog(@"%@",response.etag);
}];
```

7.8 断点续传下载

当下载大对象到本地文件时，经常出现因网络不稳定或程序崩溃导致下载失败的情况。失败后再次重新下载不仅浪费资源，而且当网络不稳定时仍然有下载失败的风险。断点续传下载接口能有效地解决此类问题引起的下载失败，其原理是将待下载的对象分成若干个分段分别下载，并实时地将每段下载结果统一记录在checkpoint文件中，仅当所有分段都下载成功时返回下载成功的结果，否则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 iOS SDK对应方法
bucketName	桶名，必选参数。	request.bucketName
objectKey	对象名，必选参数。	request.objectKey
downloadFilePath	下载对象的本地文件全路径。	request.downloadFilePath
versionID	对象的版本号。	request.versionID
enableCheckpoint	是否开启断点续传模式，默认为NO，表示不开启。	request.enableCheckpoint
enableMD5Check	是否开启MD5校验。	request.enableMD5Check
enableForceOverwrite	是否开启强制覆盖。	request.enableForceOverwrite
checkpointFilePath	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。	request.checkpointFilePath
partSize	分段大小，单位字节，取值范围是5MB~5GB。	request.partSize
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。。	request.ifModifiedSince
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。	request.ifUnmodifiedSince
ifETagMatch	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。	request.ifETagMatch

参数	作用	OBS iOS SDK对应方法
ifETagNoneMatch	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。	request.ifETagNoneMatch

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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https://support.huaweicloud.com/intl/zh-cn/usermanual-ca/ca_01_0003.html
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 存储路径
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"filename"];
// 最大并发数
self.client.configuration.maxConcurrentDownloadRequestCount = 5;
// 断点续传下载
OBSDownloadFileRequest *request = [[OBSDownloadFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" downloadFilePath:outfilePath];

// 是否打开强制覆盖
request.enableForceOverwrite = YES;
// 分段大小
request.partSize = [NSNumber numberWithInteger:5*1024*1024];
// 是否开启断点续传
request.enableCheckpoint = YES;

request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);

};
OBSBFTask  *task = [client downloadFile:request completionHandler:^(OBSDownloadFileResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];

[task waitUntilFinished];

if(task.error){
    // 重新下载
}
```

8 管理对象

8.1 获取对象属性

您可以通过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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取对象属性
OBSGetObjectMetaDataRequest *request = [[OBSGetObjectMetaDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];


[client getObjectMetaData:request completionHandler:^(OBSGetObjectMetaDataResponse *response, NSError *error){
    NSLog(@"meta:%@\n storageClass:%@\n websiteRedirectlocation:%@\n size:%@",response.metaDataDict,response.storageClass,response.websiteRedirectLocation,response.size);
}];
```

8.2 管理对象访问权限

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OBSSetObjectACLRequest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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 上传文件
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:_imagePath];

//设置对象为公共读写
request.objectACLPolicy = OBSACLPolicyPublicRead;


request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];
```



```
// 初始化身份验证
 OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

//直接设置对象的访问权限
// 初始化拥有者对象
OBSUser *owner = [[OBSUser alloc] initWithID:@"owner id"];
// 授权用户
OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"grantee id"];
// 访问权限
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
for(int i=0;i<=20;i++){
    [policy.accessControlList addObject:grant];
}

// 设置对象访问权限
OBSSetObjectACLRequest *request = [[OBSSetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" accessControlPolicy:policy];
[client setObjectACL:request completionHandler:^(OBSSetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取对象ACL
OBSGetObjectACLRequest *request = [[OBSGetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];
```

```
[client getObjectACL:request completionHandler:^(OBSGetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

8.3 列举对象

您可以通过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 iOS SDK对应方法
bucketName	桶名。	request.bucketName
prefix	限定返回的对象名必须带有prefix前缀。	request.prefix
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。	request.marker
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。	request.maxKeys
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。	request.delimiter

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


// 初始化身份验证
```

```
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

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response.isTruncated为YES表明本次没有返回全部对象，并可通过response.nextMarker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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
// 只列举一个对象
request.maxKeys = [NSNumber numberWithInt:1];

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

指定前缀列举

以下代码展示如何指定前缀列举对象：


```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];
// 指定列出前缀为testdir2的对象
request.prefix = @"/testdir2/";

[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

//设置列举对象名字典序在"test"之后的对象
request.marker = @"test";
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
```



```
    }
}];
```

分页列举全部对象

以下代码展示分页列举全部对象：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

__block OBSListObjectsResponse *result;

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.maxKeys = [NSNumber numberWithInt:1];
// 列举全部对象
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
        result = response;
     for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
     }
        request.marker = result.nextMarker;
        dispatch_semaphore_signal(sema);
   }];
   dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} 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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];
```



```
// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.prefix = @"file/";

__block OBSListObjectsResponse *result;
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
        result = response;for (int i =0; i<response.contentsList.count; i++) {
            NSLog(@"%@ \n",response.contentsList[i].key);
        }
        request.marker = result.nextMarker;
        dispatch_semaphore_signal(sema);
    }];
    dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);} while (result.isTruncated);
```

列举根目录下所有文件夹

以下代码展示如何列举根目录下的所有文件夹信息

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// 列举根目录中的文件夹
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    for (int i =0; i<response.commonPrefixesList.count; i++) {
        NSLog(@"%@ \n",response.commonPrefixesList[i].prefix);
    }

}];
```


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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举对象
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// 根目录中的对象
[client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
    [self listObjectsByPrefix:client request:request result:response];
    for (int i =0; i<response.contentsList.count; i++) {
        NSLog(@"%@ \n",response.contentsList[i].key);
    }

}];
```

listObjectsByPrefix函数：

```
-(void) listObjectsByPrefix:(OBSClient*) client request:(OBSListObjectsRequest *) request result:(OBSListObjectsResponse*) result{

    for (OBSCommonPrefix *prefix in result.commonPrefixesList){
        request.prefix = prefix.prefix;

        [client listObjects:request completionHandler:^(OBSListObjectsResponse *response, NSError *error) {
            // 异步变同步
            dispatch_semaphore_t sema = dispatch_semaphore_create(0);
            NSLog(@"文件夹[%@]中的对象:",prefix.prefix);
            for (int i=0; i<response.contentsList.count; i++) {
                NSLog(@"%@ \n",response.contentsList[i].key);
            }
            dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
            [self listObjectsByPrefix:client request:request result:response];
        }];
    }

}
```


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“/”。
- result.commonPrefixesList包含的是文件夹的子文件夹。
- listObjectsByPrefix函数需要在头文件中导入OBSListObjectsModel.h。

8.4 删除对象

删除单个对象

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 删除对象

OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname" objectKey:@"objectname"];

[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

批量删除对象

您可以通过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对象：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 批量删除对象
OBSDeleteObjectsRequest *deleteRequest = [[OBSDeleteObjectsRequest alloc] initWithBucketName:@"bucketname"];

// 被删除对象列表
OBSObjectToDelete *object1 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname1"];
OBSObjectToDelete *object2 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname2"];

deleteRequest.objectList = @[object1,object2];

[client deleteObjects:deleteRequest completionHandler:^(OBSDeleteObjectsResponse *response, NSError *error) {
   for(int i=0;i<response.deletedList.count;i++){
      NSLog(@"%@\n",response.deletedList[i].key);
   }
}];
```

8.5 复制对象

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。

简单复制

以下代码展示了如何进行简单复制：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
```

```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-
bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname"
dstObjectKey:@"objectname2"];

// 属性重写
request.dstObjectMetaDirective = OBSMetaDirectiveCopy;

request.dstObjectStorageClass = OBSStorageClassStandard;

request.dstObjectWebsiteRedirectLocation = @"URL";

request.customContentType = @"video/mp4";


[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

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 iOS SDK对应枚举值
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。	cpSrcIfETagMatch
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。	cpSrcIfETagNoneMatch
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。	cpSrcIfModifiedSince
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。。	cpSrcIfUnmodifiedSince

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。
- 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:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname" dstObjectKey:@"objectname2"];


request.cpSrcIfETagNoneMatch = @"\"f807071206c05630b4d3c92aae4f4448\"";

request.cpSrcIfModifiedSince = @"Sunday, 06-Nov-94 08:49:37 GMT";
```



```
//request.cpSrcIfModifiedSince = [[OBSUtils      getDateFormatterRFC1123]dateFromString:@"Mon, 18 Dec 2017 03:50:49 GMT"];

[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname" dstObjectKey:@"objectname2"];

// 设置权限为完全控制权限
request.dstObjectACLPolicy = OBSACLFull_Control;
[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```


9 临时授权访问

9.1 使用临时 URL 进行授权访问

临时授权访问是指通过访问密钥、请求方法类型、请求参数等信息生成一个临时访问权限的URL，这个URL中会包含鉴权信息，您可以使用该URL进行访问OBS服务进行特定操作。在生成URL时，您需要指定URL的有效期。所有继承OBSBaseRequest的子类都能使用临时鉴权访问。

临时授权访问支持的操作以及相关信息见下表：

操作名	OBS iOS SDK类名
创建桶	OBSCreateBucketRequest
获取桶列表	OBSListBucketsRequest
删除桶	OBSDeleteBucketRequest
列举桶内对象	OBSListObjectsRequest
列举桶内多版本对象	OBSListObjectsVersionsRequest
列举分段上传任务	OBSListMultipartUploadsRequest
获取桶元数据	OBSGetBucketMetaDataRequest
获取桶区域位置	OBSGetBucketMetaDataRequest
获取桶存量信息	OBSGetBucketStorageInfoRequest
设置桶配额	OBSSetBucketQuotaRequest
获取桶配额	OBSGetBucketQuotaRequest
设置桶访问权限	OBSSetBucketACLWithCannedACLRequest、OBSSetBucketACLWithPolicyRequest
获取桶访问权限	OBSGetBucketACLRequest
开启/关闭桶日志	OBSSetBucketLoggingRequest

操作名	OBS iOS SDK类名
查看桶日志	OBSGetBucketLoggingRequest
设置桶策略	OBSSetBucketPolicyRequest、OBSSetBucketPolicyWithStringRequest
查看桶策略	OBSGetBucketPolicyRequest
删除桶策略	OBSDeleteBucketPolicyRequest
设置生命周期规则	OBSSetBucketLifecycleRequest
查看生命周期规则	OBSGetBucketLifecycleRequest
删除生命周期规则	OBSDeleteBucketLifecycleRequest
设置托管配置	OBSSetBucketWebsiteRequest
查看托管配置	OBSGetBucketWebsiteRequest
清除托管配置	OBSDeleteBucketWebsiteRequest
设置桶多版本状态	OBSSetBucketVersioningRequest
查看桶多版本状态	OBSGetBucketVersioningRequest
设置跨域规则	OBSSetBucketCORSRequest
查看跨域规则	OBSGetBucketCORSRequest
删除跨域规则	OBSDeleteBucketCORSRequest
OPTIONS桶	OBSOptionsBucketRequest
设置桶标签	OBSSetBucketTaggingRequest
查看桶标签	OBSGetBucketTaggingRequest
删除桶标签	OBSDeleteBucketTaggingRequest
上传对象	OBSPutObjectWithDataRequest、OBSPutObjectWithFileRequest
追上上传	OBSAppendObjectWithFileRequest
下载对象	OBSGetObjectToDataRequest
复制对象	OBSCopyObjectRequest
删除对象	OBSDeleteObjectRequest
批量删除对象	OBSDeleteObjectsRequest
获取对象属性	OBSGetObjectMetaDataRequest
设置对象访问权限	OBSSetObjectACLRequest
查看对象访问权限	OBSGetObjectACLRequest
初始化分段上传任务	OBSInitiateMultipartUploadRequest

操作名	OBS iOS SDK类名
上传段	OBSUploadPartWithDataRequest
复制段	OBSCopyPartRequest
列举已上传的段	OBSListPartsRequest
合并段	OBSCompleteMultipartUploadRequest
取消分段上传任务	OBSAbortMultipartUploadRequest
OPTIONS对象	OBSOptionsObjectRequest
恢复归档存储对象	OBSRestoreObjectRequest

⚠ 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。

您可以通过createV2PreSignedURL生成授权访问的临时URL。以下代码展示了如何生成常用操作的URL：

列举对象

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
OBSListObjectsRequest *request = [[OBSListObjectsRequest alloc] initWithBucketName:@"bucketname"];

// V2生成授权访问url
[client createV2PreSignedURL:request expireAfter:3600 completionHandler:^(NSString *urlString, NSString *httpVerb, NSDictionary *signedHeaders) {
    NSLog(@"%@",urlString);
}]
```

获取对象

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest alloc] initWithBucketName:@"bucketname" objectKey:@"objectkey"];

// V2生成授权访问url
[client createV2PreSignedURL:request expireAfter:3600 completionHandler:^(NSString *urlString, NSString *httpVerb, NSDictionary *signedHeaders) {
    NSLog(@"%@",urlString);
}]
```

10 多版本控制

10.1 多版本控制简介

OBS支持保存一个对象的多个版本，使您更方便地检索和还原各个版本，在意外操作或应用程序故障时快速恢复数据。

更多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

您可以通过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 iOS 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（OBSListObjectsRequest）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（OBSListObjectsVersionsRequest）。 5. 除了删除标记外，每个版本的对象存储均需计费	OBSVersioningStatusEnabled

多版本状态	说明	OBS iOS SDK对应值
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。	OBSVersioningStatusSuspended

以下代码展示了如何设置桶的多版本状态：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 配置桶多版本
OBSBucketVersioningConfiguration *conf1 = [OBSBucketVersioningConfiguration new];

conf1.status = OBSVersioningStatusEnabled;
// 设置桶多版本
OBSSetBucketVersioningRequest *request = [[OBSSetBucketVersioningRequest alloc] initWithBucketName:@"bucketname" configuration: conf];
[client setBucketVersioning:request completionHandler:^(OBSSetBucketVersioningResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

10.3 查看桶多版本状态

您可以通过getBucketVersioning查看桶的多版本状态。以下代码展示了如何查看桶的多版本状态：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 查看桶多版本状态
OBSGetBucketVersioningRequest *request = [[OBSGetBucketVersioningRequest alloc] initWithBucketName:@"bucketname"];
[client getBucketVersioning:request completionHandler:^(OBSGetBucketVersioningResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

10.4 获取多版本对象

您可以通过OBSAbstractGetObjectRequest的子类，通过设置request.versionID来获取多版本对象，示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取多版本对象
OBSGetObjectToDataRequest *request = [[OBSGetObjectToDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];

// 多版本ID
request.versionID = @"";

// 下载进度
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t totalBytesExpectedToWrite) {
```

```
    NSLog(@"%0.1f%%",(float)(totalBytesWritten)*100/(float)totalBytesExpectedToWrite);
};
// 下载的数据
__block NSMutableData *objectData = [NSMutableData new];
request.onReceiveDataBlock = ^(NSData *data) {
    [objectData appendData:data];
};

[ client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

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

您可以通过OBSCopyObjectRequest接口传入版本号（versionID）来复制多版本对象，示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSCopyObjectRequest *request = [[OBSCopyObjectRequest alloc]initWithSrcBucketName:@"source-bucketname" srcObjectKey:@"objectname1" dstBucketName:@"destination-bucketname" dstObjectKey:@"objectname2"];

// 被拷贝对象多版本ID
request.srcObjectVersionID = @"testVersionID";
[client copyObject:request completionHandler:^(OBSCopyObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```

10.6 恢复多版本归档存储对象

您可以通过OBSRestoreObjectRequest接口传入版本号（versionID）来恢复多版本归档存储对象，示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
// 恢复归档存储对象
OBSRestoreObjectRequest *request = [[OBSRestoreObjectRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" storeDays:[NSNumber numberWithInt:30]];//1 to 30
request.restoreTier = OBSRestoreTierExpedited;

//多版本号
request.versionID = @"多版本ID";
[client restoreObject:request completionHandler:^(OBSRestoreObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

⚠ 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。产生的标准存储类别的对象副本有效期将会延长，并且收取延长时间段产生的标准存储副本费用。

10.7 列举多版本对象

您可以通过OBSListObjectsVersionsRequest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。
- OBSListObjectsVersionsRequest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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
 OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc] initWithBucketName:@"bucketname"];

[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError *error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
   }
 }];
```

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response.isTruncated为YES表明本次没有返回全部对象，并可通过response.nextKeyMarker和response.versionIdMarker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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
```



```
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc] initWithBucketName:@"bucketname"];

// 最大列举100条
request.maxKeys = [NSNumber numberWithInt:100];


[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError *error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
  }
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc] initWithBucketName:@"bucketname"];

// 指定前缀列举
request.prefix = @"/";
```



```
[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
   }

}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

// 指定位置列举
request.keyMarker = @"/";

[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {
    for (int i =0; i<response.versionList.count; i++) {
       NSLog(@"%@ \n",response.versionList[i].key);
   }
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```

```
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象

__block OBSListObjectsVersionsResponse *result;

    // 列举对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
initWithBucketName:@"bucketname"];

request.maxKeys = [NSNumber numberWithInt:1];
// 列举全部对象
do {
    dispatch_semaphore_t sema = dispatch_semaphore_create(0);
    [client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response,
NSError *error) {
        result = response;
     for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }
        request.keyMarker = result.nextKeyMarker;
        dispatch_semaphore_signal(sema);
  }];
   dispatch_semaphore_wait(sema, DISPATCH_TIME_FOREVER);
} 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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 列举多版本对象

__block OBSListObjectsVersionsResponse *result;

// 列举对象
OBSListObjectsVersionsRequest *request = [[OBSListObjectsVersionsRequest alloc]
```

```
initWithBucketName:@"bucketname"];

request.delimiter = @"/";
// 根目录中的对象
[client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response, NSError
*error) {

    for (int i =0; i<response.versionList.count; i++) {
        NSLog(@"%@ \n",response.versionList[i].key);
    }
    [self listVersionObjectsByPrefix:client request:request result:response];
}];
```

listVersionObjectsByPrefix函数：

```
-(void) listVersionObjectsByPrefix:(OBSClient*) client request:(OBSListObjectsVersionsRequest *) request
result:            (OBSListObjectsVersionsResponse*) result{

    for (NSString * prefix in result.commonPrefixesList){
        NSLog(@"文件夹[%@]中的对象:",prefix);
        request.prefix = prefix;
        [client listObjectsVersions:request completionHandler:^(OBSListObjectsVersionsResponse *response,
NSError *error) {
            for (int i =0; i<response.versionList.count; i++) {
                NSLog(@"%@ \n",response.versionList[i].key);
            }
            [self listVersionObjectsByPrefix:client request:request result:response];
        }];
    }

}
```

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

设置多版本对象访问权限

您可以通过OBSSetObjectACLRequest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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
```

```
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置多版本对象访问策略
OBSUser *owner = [[OBSUser alloc] initWithID:@"249e6c2bfbc74c928d5893895543029e"];

OBSACLGranteeUser *grantee = [[OBSACLGranteeUser alloc]initWithID:@"AKjdsjklSKLL/
DSKDSLADLADLjdsjald231124"];
// 为授权用户设置所有权限
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];
OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
for(int i=0;i<=20;i++){
    [policy.accessControlList addObject:grant];
 }

OBSSetObjectACLRequest *request = [[OBSSetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" accessControlPolicy:policy];

request.versionID = @"多版本ID";

[client setObjectACL:request completionHandler:^(OBSSetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

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OBSGetObjectACLRequest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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取多版本对象访问策略
OBSGetObjectACLRequest *request = [[OBSGetObjectACLRequest
alloc]initWithBucketName:@"bucketname" objectKey:@"objectname"];
// 设置多版本ID
request.versionID = @"多版本ID";
```



```
[client getObjectACL:request completionHandler:^(OBSGetObjectACLResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

10.9 删除多版本对象

删除单个多版本对象

您可以通过OBSDeleteObjectRequest接口传入版本号（versionID）删除多版本对象，示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 删除对象
OBSDeleteObjectRequest *request = [[OBSDeleteObjectRequest alloc] initWithBucketName:@"bucketname" objectKey:@"objectname"];

// 设置多版本ID
request.versionID = @"versionID";

[client deleteObject:request completionHandler:^(OBSDeleteObjectResponse *response, NSError *error) {
    NSLog(@"%@",response);
}];
```

批量删除多版本对象

您可以通过OBSDeleteObjectsRequest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，示例代码如下：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];
```



```
//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

 // 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 批量删除对象
OBSDeleteObjectsRequest *deleteRequest = [[OBSDeleteObjectsRequest alloc]
initWithBucketName:@"bucketname"];

// 多版本删除对象
OBSObjectToDelete *object = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname"
versionID:@"versionID"];
OBSObjectToDelete *object1 = [[OBSObjectToDelete alloc] initWithObjectKey:@"objectname"
versionID:@"versionID"];

deleteRequest.objectList = @[object,object1];

[client deleteObjects:deleteRequest completionHandler:^(OBSDeleteObjectsResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```


11 生命周期管理

11.1 生命周期管理简介

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。
- 桶必须开启多版本状态，历史版本对象转换策略和历史版本对象过期时间配置才能生效。

11.2 设置生命周期规则

您可以通过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：

```
 static OBSClient *client;
 NSString *endPoint = @"your-endpoint";

// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置一条规则
OBSLifecycleRule *rule = [[OBSLifecycleRule alloc] initWithID:@"delete-2-days" prefix:@"test/" status:OBSLifecycleStatusEnabled];
// 创建30天后转换成低频访问存储
OBSLifecycleTransition* transitionStandard = [[OBSLifecycleTransition alloc]initWithDays:[NSNumber numberWithInteger:30] storageClass:OBSStorageClassStandardIA];
// 创建60天后转换成指定存储
OBSLifecycleTransition* transitionGlacier= [[OBSLifecycleTransition alloc]initWithDays:[NSNumber numberWithInteger:60] storageClass:OBSStorageClassGlacier];
// 30天后历史版本转换成标准存储
OBSLifecycleNoncurrentVersionTransition* noncurrentTransistionStandard = [[OBSLifecycleNoncurrentVersionTransition alloc]initWithDays:[NSNumber numberWithInteger:30] storageClass:OBSStorageClassStandardIA];
//60天后历史版本转换成归档存储
OBSLifecycleNoncurrentVersionTransition* noncurrentTransistionGlacier= [[OBSLifecycleNoncurrentVersionTransition alloc]initWithDays:[NSNumber numberWithInteger:60] storageClass:OBSStorageClassGlacier];


[rule.transitionList addObject:transitionStandard];
[rule.transitionList addObject:transitionGlacier];

[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionStandard];
[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionGlacier];

rule.expiration = expiration;
rule.noncurrentVersionExpiration  = noncurrentExpiration;

OBSSetBucketLifecycleRequest *request = [[OBSSetBucketLifecycleRequest alloc]initWithBucketName:@"bucketname" ];
[request.lifecycleRuleList addObject: rule];
OBSLifecycleRule* rule2 = [rule copy];
```



```
rule2.ID = @"123";
rule2.prefix = @"test1/";
[request.lifecycleRuleList addObject: rule2];

 [client setBucketLifecycle:request completionHandler:^(OBSSetBucketLifecycleResponse *response, NSError
*error){
       NSLog(@"%@",response);
 }];
```

设置过期时间

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 满足的前缀对象
OBSLifecycleRule *rule = [[OBSLifecycleRule alloc] initWithID:@"delete-2-days" prefix:@"test/"
status:OBSLifecycleStatusEnabled];

// 新版本过期时间
OBSLifecycleExpiration* expiration = [[OBSLifecycleExpiration alloc]initWithDays:[NSNumber
numberWithInteger:61]];
// 历史版本过期时间
OBSLifecycleNoncurrentVersionExpiration* noncurrentExpiration =
[[OBSLifecycleNoncurrentVersionExpiration alloc]initWithDays:[NSNumber numberWithInteger:61]];

[rule.transitionList addObject:transitionStandard];
[rule.transitionList addObject:transitionGlacier];

[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionStandard];
[rule.noncurrentVersionTransitionList addObject:noncurrentTransistionGlacier];

rule.expiration = expiration;
rule.noncurrentVersionExpiration  = noncurrentExpiration;

OBSSetBucketLifecycleRequest *request = [[OBSSetBucketLifecycleRequest
alloc]initWithBucketName:@"bucketname" ];
[request.lifecycleRuleList addObject: rule];
OBSLifecycleRule* rule2 = [rule copy];
rule2.ID = @"123";
rule2.prefix = @"test1/";
[request.lifecycleRuleList addObject: rule2];

[client setBucketLifecycle:request completionHandler:^(OBSSetBucketLifecycleResponse *response, NSError
*error){
   NSLog(@"%@",response);
}];
```


11.3 查看生命周期规则

您可以通过getBucketLifecycle查看桶的生命周期规则，以下代码展示了如何查看桶的生命周期规则：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
// 查看生命周期
OBSGetBucketLifecycleRequest *request = [[OBSGetBucketLifecycleRequest alloc] initWithBucketName:@"bucketname"];
[client getBucketLifecycle:request completionHandler:^(OBSGetBucketLifecycleResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

11.4 删除生命周期规则

您可以通过deleteBucketLifecycle查看桶的生命周期规则，以下代码展示了如何查看桶的生命周期规则：

```
 static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
// 删除生命周期
OBSDeleteBucketLifecycleRequest *request = [[OBSDeleteBucketLifecycleRequest alloc] initWithBucketName:@"bucketname"];
```



```
[client deleteBucketLifecycle:request completionHandler:^(OBSDeleteBucketLifecycleResponse *response,
NSError *error){
        NSLog(@"%@",response);
}];
```

12 跨域资源共享

12.1 跨域资源共享简介

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

您可以通过setBucketCORS设置桶的跨域规则，如果原规则存在则覆盖原规则。以下代码展示了如何设置跨域规则：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

 //初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置桶的跨域资源共享

OBSSetBucketCORSRequest *request = [[OBSSetBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];
OBSCORSRule* rule = [OBSCORSRule new];
// 指定允许的跨域请求方法(GET/PUT/DELETE/POST/HEAD)
rule.allowedMethodList =
@[OBSCORSHTTPGET,OBSCORSHTTPPUT,OBSCORSHTTPPOST,OBSCORSHTTPHEAD];
```

```
// 指定允许跨域请求的来源
rule.allowedOriginList = @[@"www.example1.com",@"www.example2.com"];
// 允许的header
rule.allowedHeaderList = @[@"allowedheader1",@"allowedheader2"];
// 指定允许用户从应用程序中访问的header
rule.exposeHeaderList = @[@"exposeheader_1",@"exposeheader_2"];
// 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.maxAgeSeconds = [NSNumber numberWithInt:100];
[request.bucketCORSRuleList addObject:rule];
[client setBucketCORS:request completionHandler:^(OBSSetBucketCORSResponse *response, NSError *error)
{
    NSLog(@"%@",response);
 }];
```

12.3 查看跨域规则

您可以通过getBucketCORS查看桶的跨域规则。以下代码展示了如何查看跨域规则：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶的跨域资源配置
OBSGetBucketCORSRequest *request = [[OBSGetBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];

[client getBucketCORS:request completionHandler:^(OBSGetBucketCORSResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

12.4 删除跨域规则

您可以通过deleteBucketCORS删除桶的跨域规则。以下代码展示了如何删除跨域规则：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
```



```
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 删除桶的跨域资源配置
OBSDeleteBucketCORSRequest *request = [[OBSDeleteBucketCORSRequest alloc]
initWithBucketName:@"bucketname"];

[client deleteBucketCORS:request completionHandler:^(OBSDeleteBucketCORSResponse *response, NSError
*error) {
    NSLog(@"%@",response);
}];
```

13 设置访问日志

13.1 日志简介

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。

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

您可以通过setBucketLogging开启桶日志功能。

说明

日志目标桶与源桶必须在同一个区域（region）。

已支持日志目标桶的存储类型：低频访问存储或归档存储或标准存储。

开启桶日志

以下代码展示了如何开启桶日志：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
```

```
client = [[OBSClient alloc] initWithConfiguration:conf];

//设置桶访问日志
// 第一步 设置目标桶的日志投递组访问权限
OBSUser *owner = [[OBSUser alloc] initWithID:@"ownerID"];

OBSACLGranteeLogDelivery *grantee = [OBSACLGranteeLogDelivery new];
OBSACLGrant *grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

// 设置目标桶的日志投递组为完全控制权限
OBSACLGranteeUser *userGrantee = [[OBSACLGranteeUser alloc]initWithID:@"granteeID"];
OBSACLGrant *userGrant = [[OBSACLGrant alloc]initWithGrantee:userGrantee
permission:OBSACLFull_Control];

OBSACLGranteeAllUsers *alluserGrantee = [OBSACLGranteeAllUsers new];
OBSACLGrant *alluserGrant = [[OBSACLGrant alloc]initWithGrantee:alluserGrantee
permission:OBSACLFull_Control];

OBSAccessControlPolicy *policy = [OBSAccessControlPolicy new];
policy.owner = owner;
[policy.accessControlList addObject:grant];
[policy.accessControlList addObject:userGrant];

OBSSetBucketACLWithPolicyRequest *setACLRequest = [[OBSSetBucketACLWithPolicyRequest
alloc]initWithBucketName:@"bucketname" accessControlPolicy:policy];

[client setBucketACL:setACLRequest completionHandler:^(OBSSetBucketACLResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];


// 第二步 设置桶的日志管理配置
grant = [[OBSACLGrant alloc]initWithGrantee:grantee permission:OBSACLFull_Control];

OBSSetBucketLoggingRequest *request = [[OBSSetBucketLoggingRequest
alloc]initWithBucketName:@"bucketname"];

OBSLoggingEnabled* enabledItem = [[OBSLoggingEnabled alloc]initWithTargetBucket:@"bucketname"
targetPrefix:@"access-log"];

[enabledItem.targetGrantsList addObject:userGrant];
[enabledItem.targetGrantsList addObject:alluserGrant];

[request.loggingEnabledList addObject:enabledItem];
[client setBucketLogging:request completionHandler:^(OBSSetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

13.3 查看桶日志配置

您可以通过getBucketLogging查看桶日志配置。以下代码展示了如何查看桶日志配置：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];
```



```
// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶访问日志
OBSGetBucketLoggingRequest *request = [[OBSGetBucketLoggingRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketLogging:request completionHandler:^(OBSGetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

13.4 关闭桶日志

关闭桶日志功能实际上就是调用setBucketLogging将日志配置清空，以下代码展示了如何关闭桶日志：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

OBSSetBucketLoggingRequest *request = [[OBSSetBucketLoggingRequest
alloc]initWithBucketName:@"bucketname"];

[client setBucketLogging:request completionHandler:^(OBSSetBucketLoggingResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

14 静态网站托管

14.1 静态网站托管简介

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 设置托管配置

您可以通过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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];
```

```
OBSWebsiteConfCustom* redirectCustom = [OBSWebsiteConfCustom new];

// 配置默认主页
OBSWebsiteConfCustomIndexDocument* indexDocument = [[OBSWebsiteConfCustomIndexDocument
alloc]initWithSuffix:@"index.html"];
// 配置默认错误界面
OBSWebsiteConfCustomErrorDocument* errorDocument = [[OBSWebsiteConfCustomErrorDocument
alloc]initWithKey:@"Error.html"];

OBSWebsiteConfCustomRoutingRule* redirectRule = [OBSWebsiteConfCustomRoutingRule new];

OBSWebsiteConfCustomCondition* condition = [OBSWebsiteConfCustomCondition new];
condition.keyPrefixEquals = @"docs/";
//    condition.httpErrorCodeReturnedEquals = @"404";

//  配置重定向规则
OBSWebsiteConfCustomRedirect* redirect = [OBSWebsiteConfCustomRedirect new];
redirect.replaceKeyPrefixWith = @"documents/";
redirect.protocol = @"http";
redirect.hostName = @"URL";
//    redirect.replaceKeyWith = @"error.html";
redirect.httpRedirectCode = @"301";


redirectRule.condition = condition;
redirectRule.redirect = redirect;

[redirectCustom.indexDocumentList addObject:indexDocument];
[redirectCustom.errorDocumentList addObject:errorDocument];
[redirectCustom.routingRulesList addObject:redirectRule];

OBSSetBucketWebsiteRequest *request = [[OBSSetBucketWebsiteRequest
alloc]initWithBucketName:@"bucketname" configuration:redirectCustom];

[client setBucketWebsite:request completionHandler:^(OBSSetBucketWebsiteResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 配置所有请求重定向
```

```
OBSWebsiteConfRedirectAll* redirectAll = [OBSWebsiteConfRedirectAll new];
redirectAll.hostName = @"URL";
OBSSetBucketWebsiteRequest *request = [[OBSSetBucketWebsiteRequest
alloc]initWithBucketName:@"bucketname" configuration:redirectAll];

 [client setBucketWebsite:request completionHandler:^(OBSSetBucketWebsiteResponse *response, NSError
*error){
    NSLog(@"%@",response);
 }];
```

14.3 查看托管配置

您可以通过getBucketWebsite查看桶的托管配置。以下代码展示了如何查看托管配置：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取托管配置
OBSGetBucketWebsiteRequest *request = [[OBSGetBucketWebsiteRequest alloc]
initWithBucketName:@"bucketname"];
[client getBucketWebsite:request completionHandler:^(OBSGetBucketWebsiteResponse *response, NSError
*error){
    NSLog(@"%@",response);
}];
```

14.4 清除托管配置

您可以通过deleteBucketWebsite清除桶的托管配置。以下代码展示了如何清除托管配置：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
```

```
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 清除托管配置
OBSDeleteBucketWebsiteRequest *request = [[OBSDeleteBucketWebsiteRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucketWebsite:request completionHandler:^(OBSDeleteBucketWebsiteResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```

15 标签管理

15.1 标签简介

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

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 设置桶标签
OBSSetBucketTaggingRequest *request = [[OBSSetBucketTaggingRequest alloc]initWithBucketName:@"bucketname"];
[request.tagList addObject:[[OBSBucketTag alloc]initWithKey:@"tagkey" value:@"tagvalue"]];
[request.tagList addObject:[[OBSBucketTag alloc]initWithKey:@"tagkey1" value:@"tagvalue1"]];

[client setBucketTagging:request completionHandler:^(OBSSetBucketTaggingResponse *response, NSError *error){
    NSLog(@"%@",response);
 }];
```


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

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 获取桶标签
OBSGetBucketTaggingRequest *request = [[OBSGetBucketTaggingRequest alloc] initWithBucketName:@"bucketname"];
[client getBucketTagging:request completionHandler:^(OBSGetBucketTaggingResponse *response, NSError *error){
    NSLog(@"%@",response);
}];
```

15.4 删除桶标签

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

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];
```



```
// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 删除桶标签
OBSDeleteBucketTaggingRequest *request = [[OBSDeleteBucketTaggingRequest alloc]
initWithBucketName:@"bucketname"];
[client deleteBucketTagging:request completionHandler:^(OBSDeleteBucketTaggingResponse *response,
NSError *error){
    NSLog(@"%@",response);
}];
```


16 服务端加密

16.1 服务端加密简介

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

OBS iOS SDK支持服务端加密的接口见下表：

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

OBS iOS SDK接口方法	描述	支持加密类型
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

上传对象加密

以下代码展示了在上传对象时如何使用服务端加密功能：

- SSE-C加密

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];


// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// SSE-C加密上传对象
NSData *uploadData = [NSData dataWithContentsOfFile:_imagePath];
OBSPutObjectWithDataRequest *request = [[OBSPutObjectWithDataRequest alloc]initWithBucketName:@"bucketname" objectKey:@"test/image1" uploadData:uploadData];

 // 加密
request.encryption = [[OBSEncryptionTypeCustomer alloc]initWithAlgorithm:@"AES256" key:@"K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=" keyMD5:@"4XvB3tbNTN+tIEVa0/fGaQ==""];

request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)totalBytesSent*100/(float)totalBytesExpectedToSend);
};
[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response);
}] ;
```


说明

- key：密钥通过AES256加密生成。
- keyMD5：密钥通过MD5生成值，再将此值通过base64加密。

- SSE-KMS加密

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// SSE-KMS加密上传
OBSPutObjectWithFileRequest *request = [[OBSPutObjectWithFileRequest alloc]initWithBucketName:@"bucketname" objectKey:@"objectname" uploadFilePath:_imagePath];

// SSE-KMS加密
request.encryption = [[OBSEncryptionTypeKMS alloc]initWithKeyID:nil];

request.uploadProgressBlock = ^(int64_t bytesSent, int64_t totalBytesSent, int64_t totalBytesExpectedToSend) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesSent*10000/totalBytesExpectedToSend)/100);
};

[client putObject:request completionHandler:^(OBSPutObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
```



```
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 下载对象解密
NSString * outfilePath = [NSTemporaryDirectory() stringByAppendingString:@"test.png"];
OBSGetObjectToFileRequest *request = [[OBSGetObjectToFileRequest
alloc]initWithBucketName:@"bbucketname" objectKey:@"objectname" downloadFilePath:outfilePath];

// 与上传时的密钥一致
request.encryption = [[OBSEncryptionTypeCustomer alloc]initWithAlgorithm:@"AES256"
key:@"K7QkYpBkM5+hcs27fsNkUnNVaobncnLht/rCB2o/9Cw=" keyMD5:@"4XvB3tbNTN+tIEVa0/fGaQ==""];
request.downloadProgressBlock = ^(int64_t bytesWritten, int64_t totalBytesWritten, int64_t
totalBytesExpectedToWrite) {
    NSLog(@"%0.1f%%",(float)floor(totalBytesWritten*10000/totalBytesExpectedToWrite)/100);
};
[client getObject:request completionHandler:^(OBSGetObjectResponse *response, NSError *error){
    NSLog(@"%@",response.etag);
}];
```

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	用户的账号出现异常（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地址关联的账号超过了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
IllegalVersioningConfiguration Exception	请求中的版本配置无效。	400 Bad Request
IllegalLocationConstraint Exception	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
IncorrectNumberOfFilesInPost Request	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
nsufficientStorageSpace	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
InvalidBucketState	无效的桶状态。	409 Conflict

错误码	描述	HTTP状态码
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
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

错误码	描述	HTTP状态码
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
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
MaxPostPreDataLengthExceeded Error	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
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。	Not Found
NoSuchCustomDomain	请求的用户域不存在。	404 Not Found

错误码	描述	HTTP状态码
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	账号未在系统中注册，必须先在系统中注册了才能使用该账号。	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
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

错误码	描述	HTTP状态码
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与记录中任何账号的都不匹配。	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
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

错误码	描述	HTTP状态码
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

通过响应中的error来判断请求是否异常，如果error等于空则请求无异常，否则请求失败，错误信息将会在窗口中打印，代码如下所示:

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc] initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 创建一个请求
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    // 请求成功，错误为空
    if(!error){
        //处理请求
         NSLog(@"%@",response.headers);
    }
}];
```

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OBSBaseResponse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：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
```

```
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// 创建一个请求
OBSListBucketsRequest *request = [OBSListBucketsRequest new];

[client listBuckets:request completionHandler:^(OBSListBucketsResponse *response, NSError *error) {
    NSLog(@"%@",response.headers);
}];
```

17.4 日志分析

设置方式

开启系统日志记录，代码如下所示：

```
static OBSClient *client;
NSString *endPoint = @"your-endpoint";
// 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
char* ak_env = getenv("AccessKeyID");
char* sk_env = getenv("SecretAccessKey");
NSString *AK = [NSString stringWithUTF8String:ak_env];
NSString *SK = [NSString stringWithUTF8String:sk_env];

// 初始化身份验证
OBSStaticCredentialProvider *credentialProvider = [[OBSStaticCredentialProvider alloc]
initWithAccessKey:AK secretKey:SK];

//初始化服务配置
OBSServiceConfiguration *conf = [[OBSServiceConfiguration alloc] initWithURLString:endPoint
credentialProvider:credentialProvider];

// 初始化client
client = [[OBSClient alloc] initWithConfiguration:conf];

// *****日志设置*******
// 设置系统日志打印
[client addLogger:[OBSDDASLLogger sharedInstance]];
// 打开窗口打印
[client addLogger:[OBSDDTTYLogger sharedInstance]];

// 设置日志文件记录
OBSDDFileLogger *fileLogger = [[OBSDDFileLogger alloc] init];
// 保留时间
fileLogger.rollingFrequency = 60 * 60 * 24; // 24 hour rolling
// 最大文件数目
fileLogger.logFileManager.maximumNumberOfLogFiles = 7;

// 自定义日志添加
[client addLogger:fileLogger];
```



```
// 日志文件地址
OBSDDLogFileInfo *ts =[fileLogger currentLogFileInfo];
NSLog(@"%@", ts);

// 设置日志等级
[client setLogLevel:OBSDDLogLevelDebug];

 // 开启日志
[client setASLogOn];
```

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OBSDDLogLevelVerbose日志信息最丰富，OBSDDLogLevelError日志信息最少。默认为OBSDDLogLevelInfo。

类型	描述
OBSDDLogLevelOff	关闭日志。
OBSDDLogLevelError	只打印错误信息。
OBSDDLogLevelWarning	打印错误和警告信息。
OBSDDLogLevelInfo	打印错误、警告、基本运行信息。
OBSDDLogLevelDebug	打印错误、警告、基本运行、调试信息。
OBSDDLogLevelVerbose	打印所有日志信息。

18 常见问题

18.1 如何获取临时 AK/SK

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

获取临时访问密钥的接口请参考**获取临时AK/SK和securitytoken**。

使用临时访问密钥的方法请参考**临时访问密钥创建OBS客户端代码**。

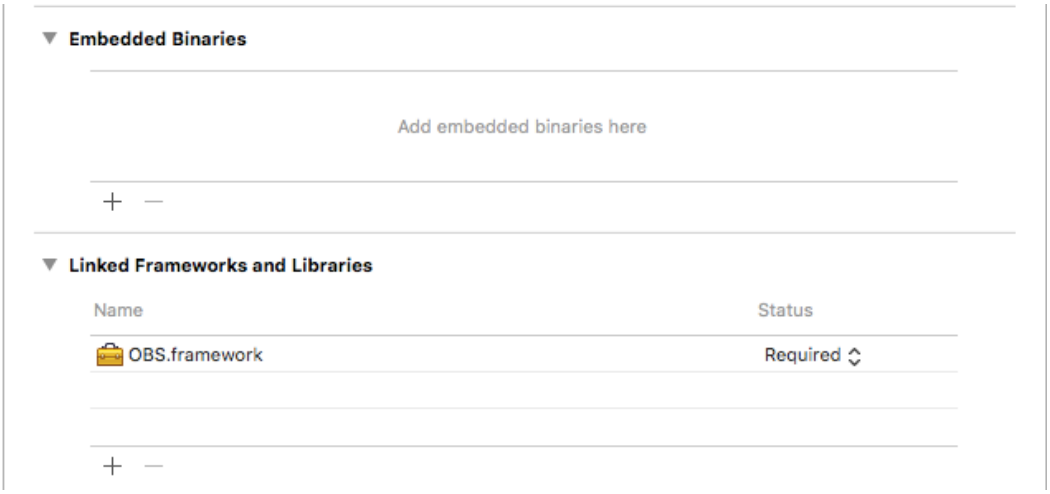
18.2 项目打包出错

打包时候报如下错误：

OBS.framework” did not contain a “archived-expanded-entitlements.xcent” resource.

修复方法：

OBS.framework为静态库，请确保*Embedded Binaries*中不包含OBS.framework



18.3 项目编译报错 duplicate symbols

将已有的Swift项目中集成OBS.framework，在Build Settings的Other Linker Flags中添加了-force_load $(SRCROOT)/XXX/OBS.framework/OBS，然后将Allow Nonmodular includes in Framework Modules设置为Yes，编译时报错：

Showing Recent Errors Only

1639 duplicate symbols

修复方法：

Xcode -> TARGETS -> Build Settings -> Other Linker Flags 添加一行"-ld64"

A API 参考

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