### **前言**

#### **SDK下载**

* Python SDK 开发包最新版本 [v1.1.0](http://pws.paic.com.cn/m)
* Python SDK 开发包历史版本 [v1.0.0](http://pws.paic.com.cn/m)

#### **简介**

* OBS Python SDK 适用于 Python3版本
* 本文档主要介绍 OBS Python SDK 的接口使用及注意事项
* 并且假设您是平安云用户，且已经开通了OBS服务
* 如果您还没有开通或者还不了解平安云 OBS 服务，请登录[OBS产品主页](https://yun.pingan.com/)进行了解

#### **兼容性**

* 对于 v1.0.0 SDK：

接口兼容，不影响所有 v1.0.0 用户的使用。在 v1.1.0 版本：

上传文件接口新增了默认参数*object\_metadata*，以支持上传自定义用户元数据；

上传文件接口支持上传网络流；

上传文件支持分片上传；

下载文件接口支持获取自定义元数据

### **初始化服务**

ObsOperator是 OBS 服务的 Python 客户端，它为调用者提供一系列与 OBS 进行交互的接口，用于上传文件、下载文件等。用户在调用 SDK 接口前必须先初始化 ObsOperator

创建服务实例：

**from** obs **import** ObsOperator, ObjectMetadata, MultipartUploadFileRequest

obs\_host = ""

obs\_access\_key = ""  
obs\_secret\_key = ""

obs = ObsOperator(obs\_host, obs\_access\_key, obs\_secret\_key)

### **上传文件**

#### **上传字符串**

**def put\_object**(self, bucket\_name, object\_key, data, object\_metadata=**None**)

**功能描述**

以字符串为上传对象，若对象存在则更新，并支持添加自定义元数据

**入参**

*bucket\_name: 保存对象的 bucket名称*

*object\_key: 对象 key，需自行实现策略保证同一 bucket 下 object\_key 唯一, 不能为空，首字符不能为路径符*

*data: 待上传的字符串*

*object\_metadata: 可选参数，可传入自定义元数据*

**出参**

*PutObjectResult  返回对象*

**Demo**

bucket\_name = “”

object\_key = “”

upload\_content = “”

object\_metadata = ObjectMetadata()

object\_metadata.add\_user\_metadata(“key”,“value”)

ret = obs.put\_object(bucket\_name, object\_key, upload\_content，object\_metadata)

print(ret.get\_e\_tag())

#### **上传本地文件**

**def put\_object**(self, bucket\_name, object\_key, data, object\_metadata=**None**)

**功能描述**

以文件对象为上传对象，若对象存在则更新，并支持添加自定义元数据

**入参**

*bucket\_name: 保存对象的 bucket名称*

*object\_key: 对象 key，需自行实现策略保证同一 bucket 下 object\_key 唯一, 不能为空，首字符不能为路径符*

*data: 待上传的文件对象*

*object\_metadata: 可选参数，可传入自定义元数据*

**出参**

*PutObjectResult  返回对象*

**Demo1**

bucket\_name = “”

object\_key = “”

local\_file\_upload\_path = “”

**with** open(local\_file\_upload\_path, "rb") **as** from\_file:  
 ret = obs.put\_object(bucket\_name, object\_key, from\_file)

print(ret.get\_e\_tag())

**Demo2**

**使用简单接口上传本地文件**

bucket\_name = “”

object\_key = “”

local\_file\_upload\_path = “”

ret = obs.put\_object\_from\_file(bucket\_name, object\_key, local\_file\_upload\_path)

print(ret.get\_e\_tag())

#### **上传网络流**

**def put\_object** (self, bucket\_name, object\_key, data, object\_metadata=**None**)

**功能描述**

以网络流为上传对象，若对象存在则更新，并支持添加自定义元数据

**入参**

*bucket\_name: 保存对象的 bucket名称*

*object\_key: 对象 key，需自行实现策略保证同一 bucket 下 object\_key 唯一, 不能为空，首字符不能为路径符*

*data: 待上传的网络流*

*object\_metadata: 可选参数，可传入自定义元数据*

**出参**

*PutObjectResult  返回对象*

**Demo**

bucket\_name = “”

object\_key = “”

input\_stream = urlopen("http://alm.paic.com.cn/project/2008/board/")  
ret = obs.put\_object(bucket\_name, object\_key, input\_stream)

print(ret.get\_e\_tag())

#### **分片上传**

**def put\_object\_multipart**(self, upload\_file\_request):

**功能描述**

对于大文件上传，可以切分成片上传。在以下情况下，应该考虑使用分片上传：

* 文件大于 100M
* 网络环境差，上传易导致失败
* 需要支持断点续传

**入参**

*upload\_file\_request: MutilpartUploadFileRequest对象。*

*分片上传参数 upload\_file\_request的说明：*

*每个分片（除了最后一个分片）的大小必须不小于 5M，最后一个分片的大小无限制；*

*若任何一个分片（非最后一个分片）小于 5M，每个分片的上传不会失败，然而分片完成操作（complete）会失败。*

*enable\_checkpoint 用于设置是否启用断点续传。默认启用。通过调用 upload\_file\_request.set\_enable\_checkpoint(False) 关闭断点续传功能。*

*checkpoint\_file\_path 用于设置断点续传 ucp 文件的保存位置。默认与待上传文件位于同一个目录下面。若非特殊需求，不需要更改此设置。*

*upload\_notifier 用于设置分片上传的通知接收者。可以实现一个 mutilpart\_upload\_notifier，并调用 upload\_file\_request.set\_upload\_notifier 来接收分片上传的进展的通知，以达到如进度感知等目的。*

**Demo**

**def notify\_upload**(upload\_id, state, total\_parts, finish\_parts):  
 print("%s %s %s of %s" % (upload\_id, state, finish\_parts, total\_parts))

bucket\_name = “”

object\_key = “”

multipart\_upload\_file\_path = “”

multipart\_request = MultipartUploadFileRequest()  
multipart\_request.set\_bucket\_name(bucket\_name)  
multipart\_request.set\_object\_key(object\_key)  
multipart\_request.set\_upload\_file\_path(multipart\_upload\_file\_path)  
multipart\_request.set\_upload\_notifier(notify\_upload)  
  
obs.put\_object\_multipart(multipart\_request)

### **下载文件**

**def get\_object**(self, bucket\_name, object\_key)

**功能描述**

下载指定对象

**入参**

*bucket\_name: 保存对象的 bucket名称*

*object\_key: 对象 key*

**出参**

*S3Object 返回对象，可以获取对象流自己保存，也可以传入文件对象直接存储*

**Demo**

bucket\_name = “”

object\_key = “”

to\_file\_path = “”

s3\_object = obs.get\_object(bucket\_name, object\_key)

#读取对象流自己存储

**with** open(to\_file\_path, "wb") **as** to\_file:  
 shutil.copyfileobj(s3\_object.get\_object\_stream()

, to\_file)

#传入文件对象直接存储  
**with** open(to\_file\_path, "wb") **as** to\_file:  
 s3\_object.to\_file(to\_file)

#获取自定义元数据

print(s3\_object.get\_object\_metadata().get\_user\_metadata())