# 盛大云存储 C 语言 SDK 使用说明

## 目录

# 1. 综述

盛大云存储 C 语言 SDK 参照盛大云存储 REST API 文档(https://cs-console.grandcloud.cn/public/docs/GrandCloud_Storage_Developer_Guide.pdf)，为用户提供标准 C 的本地接口，通过使用该 SDK，用户可以方便的接入和访问盛大云存储服务。

# 2. 系统要求

当前版本的 C 语言 SDK 依赖于一些外部的库，用户同样需要将这些库添加到自己的工程依赖中，它们是：

- libcurl 7.25.0 download: http://curl.haxx.se/download.html
- openssl 1.0.1c download: http://www.openssl.org/source/
- libxml2 download: http://www.xmlsoft.org/downloads.html

用户需要根据自己的平台，下载、编译、安装相应的库，并将相应的头文件目录和库目录添加进编译、链接路径。

# 3. 盛大云存储的基本概念

## 3.1 AccessKey

AccessKey 由盛大云存储单独颁发。 AccessKey 在所有的操作中都会被使用，并且会以明文形式传输。用于标识用户身份。每位用户一个，不会重复。

AccessKey 通过云计算网站的云存储用户信息管理获得：http://www.grandcloud.cn（需要登录）。

## 3.2 SecretAccessKey

SecretAccessKey 也由盛大云存储颁发，SecretAccessKey 总是随同 AccessKey 一起分发，一个 AccessKey 对应一个 SecretAccessKey。

SecretAccessKey 通过云计算网站的云存储用户信息管理获得：http://www.grandcloud.cn（需要登录）。

出于安全问题的考虑，对云存储的所有的操作都需要由 SecretAccessKey 签名摘要后才能有效，摘要信息将成为请求的一部分，发送给云系统。

**任何时候 SecretAccessKey 都不应发送给盛大云存储系统。**

SecretAccessKey 涉及您存储资料的安全问题，所以请妥善保存您的 SecretAccessKey，不要泄漏给第三方。如 SecretAccessKey 发生泄漏，请立即登录盛大云计算网站，云存储用户信息管理，将原 SecretAccessKey 作废，重新生成。

## 3.3 Bucket

在用户空间内，用户根据需要可以建立不同的 Bucket。

你可以把 Bucket 想象成文件系统内的目录，在盛大云存储系统中一个用户空间内最多只能有 **100** 个 Bucket。

Bucket 命名全局唯一，也就是说所有盛大云存储的用户的 Bucket 都是不一样的。 例如有 A 用户建立了名为”aaa”的 Bucket, 此时 B 用户希望创建名字同样为”aaa”的 Bucket 将会失败。

## 3.4 Bucket 的命名规则

a) 由小写字母或数字或点号(.) 或下划线(_)或破折号(-)组合而成。
b) 开头必须是 数字或者小写字母。
c) 长度必须 大于等于 3 字节 小于等于 255 字节
d) 不能是一个 IP 地址形式。比如 192.168.1.1 这样的格式是不允许的
e) 不能以 snda 作为 Bucket 的开头
f) 如果希望以后提供 DNS 解析，则 Bucket 命名还需符合 DNS 主机名的命名规则

## 3.5 Object

Object 是盛大云存储的主要对象。用户存储的内容都以 Object 形式存储在系统里。

1 个 Object 必须存储在盛大云存储系统的某个 Bucket 内。

1 个 Object 包含了 **ObjectName**，**ObjectMetadata** 以及 **ObjectData** 3 个部分。

ObjectName 就是 Object 的名字，在同一个 Bucket 下的 ObjectName 是唯一的。

## 3.6 ObjectName 的命名规则

a) 使用 Utf-8 编码规则
b) ObejctName 的长度大于等于 1 字节小于等于 1024 字节

# 4. C-SDK 的简单使用说明

当前版本的 C 语言 SDK 以源代码的形式提供给用户，用户可以直接将本 SDK 的源代码整合到自己的工程当中。SDK 支持 windows 和 Linux 2 个平台。

## 4.1 SDK 目录说明

当前版本的 SDK 包含 3 个目录：

src 目录下包含两个目录。其中 src/sdk 为 SDK 的所有文件源码，用户可以自行进行编译和并在自己的编写的程序中调用。src/example 为调用 SDK 的简单案例，其中 snda_ecs_console.c 是一个简单的例子，该例子完成了一个简单的客户端，可完成大部分云

存储操作；其它文件是 SDK 目前支持的云存储操作的简单案例，用户可模仿它们使用 SDK。

目录 libs 内的文件是封装好的 SDK 库，包括 windows 和 linux 两个环境对应的库及简单使用案例。

目录 doc 内的文件是 SDK 的使用文档。

## 4.2 安装部署

### 4.2.1 Linux 下的使用

- 使用源码

  编译安装系统要求中提及的依赖库后，Linux 下可通过类似如下命令行编译代码：

  gcc src/sdk/*.c src/example/snda_ecs_console.c -I./src/sdk -I/usr/local/include/ -I/usr/local/ssl/include/ -I/usr/local/include/libxml2/ -L/usr/local/ssl/lib -lcurl -lxml2 -lssl -lcrypto -o snda_ecs_console

- 使用静态链接库

  Linux 下的静态链接库及使用案例在 lib/linux/snda_ecs_sdk 目录下。

### 4.2.2 Windows 下的使用

Windows 下的使用较为复杂，以下以在 VS2008 中的使用为例。

#### 4.2.2.1 使用静态链接库

- 新建 Project：选择新建【Visual C++/Win32】下的【Win32 Console Application】，在参数选择中选择创建空项目即可。
- 导入必要的库：需要导入五个库，五个库的相关资源可在 lib/windows 下找到。

  a) 配置各个依赖库的头文件：本版本的 SDK 已将所有需要的库头文件放于 lib\win32\snda_ecs_sdk\include 目录中，这里配置该目录即可。

  具体做法：选中项目，右键，选择【properties】，在【Configuration Properties/C/C++/Gerneral】中作如下配置：


图 3.1 导入 include 目录

  b) 修改 Linker 配置：SDK 所需的所有 lib 文件对应的目录为\lib\win32\snda_ecs_sdk\lib，即这里配置该目录即可。

具体做法：同 a，在 properties 中，找到【Configuration Properties/Linker/General】，将各个库对应的 lib 目录添加到 Additional Library Directory:

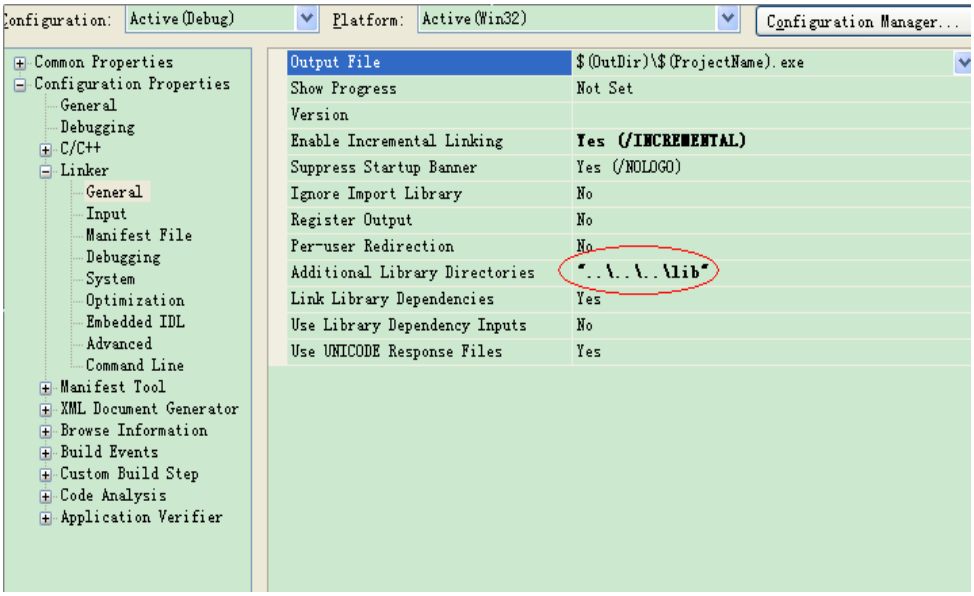


图 3.2 导入 lib 目录

c) 在【Input】的 Additional Dependencies 中加入各个 lib:

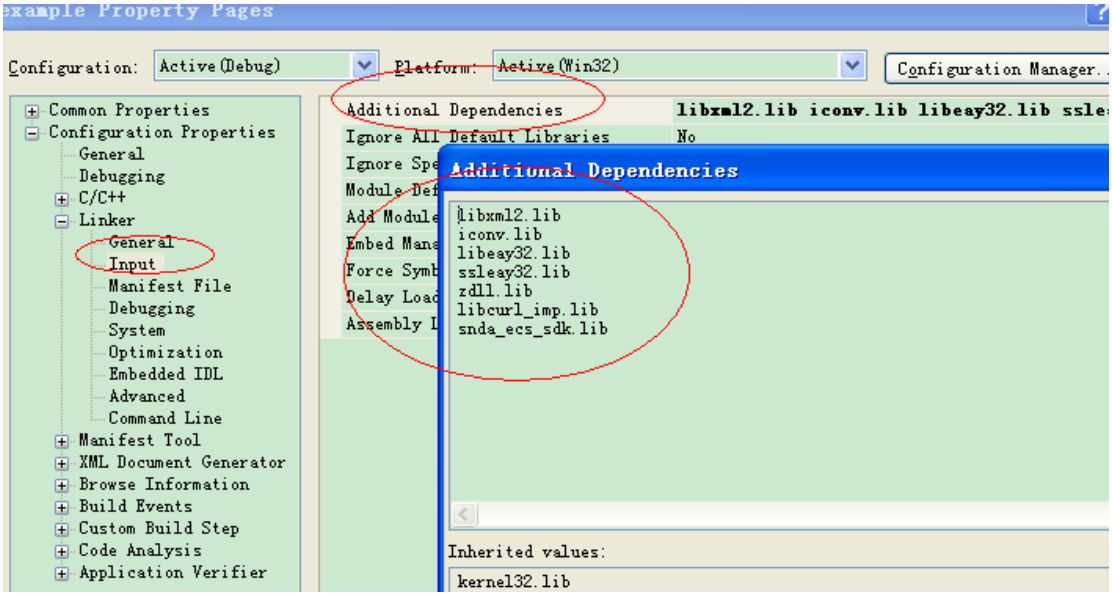


图 3.3：添加附加的 lib 库

d) 配置 dll 文件：编译项目，编译完成后，将各个库的 dll 文件拷贝至项目的 Debug 目录下即可,这里所有库的 dll 文件已存放于 lib\win32\snda_ecs_sdk\dll 目录下。以下为 Debug 目录导入的 dll 文件。

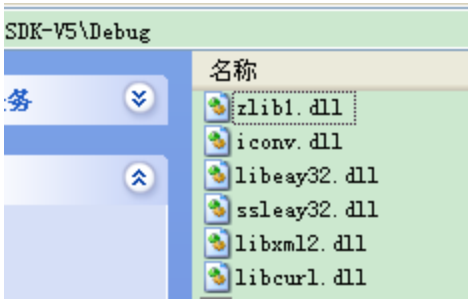


图 3.4 Debug 下的 dll 文件

- 程序中使用 SDK 时，需 include 以下头文件，即：

```
#include "snda_ecs_sdk/snda_ecs_sdk.h"
#include "snda_ecs_sdk/snda_ecs_http_util.h"
#include "snda_ecs_sdk/snda_ecs_constants.h"
#include "snda_ecs_sdk/snda_ecs_common_util.h"
```

VS2008 上的具体案例见 lib\win32\snda_ecs_sdk\vs2008\example,可以在解压 SDK 文件后，直接打开该项目即可编译运行,VS2010 用户可直接对转成 VS2010 项目使用。

#### 4.2.2.2 使用源码

本 sdk win32 的静态链接库存放于 src/sdk 下。
使用静态链接库与使用源码的配置类似，主要区别为：

- 需导入 SDK 代码：选中项目右击，选择【add】中的【 Existing Item.】，将 src/sdk 目录下的所有代码导入即可。
- 导入 lib 库时，无需要导入 snda_ecs_sdk.lib

VS2008 上的具体案例见 lib\win32\snda_ecs_sdk\vs2008\example2,可以在解压 SDK 文件后，直接打开该项目即可编译运行,VS2010 用户可直接对转成 VS2010 项目使用。

## 5. C-SDK 文档

当前版本 C 语言 SDK 代码目录结构为：

```
sdk/
|-- snda_ecs_common_util.c
|-- snda_ecs_common_util.h
|-- snda_ecs_constants.h
|-- snda_ecs_encode.c
|-- snda_ecs_encode.h
|-- snda_ecs_http_util.c
|-- snda_ecs_http_util.h
|-- snda_ecs_sdk_bucket_impl.c
|-- snda_ecs_sdk_common_impl.c
|-- snda_ecs_sdk.h
|-- snda_ecs_sdk_multipart_upload.c
|-- snda_ecs_sdk_object_impl.c
```

`-- snda_ecs_sdk_service_impl.c

## 5.1 通用说明

snda_ecs_sdk.h 用户需要且仅需要包含的头文件，其中包含：

### 5.1.1 数据结构定义

在每一种数据结构的定义之后，都紧接着定义了初始化该结构指针和释放该结构指针的方法。例如：

```
typedef struct SNDAECSHandlerError {
        char * handlererrmsg;
} SNDAECSHandlerError;

SNDAECSHandlerError* snda_ecs_init_handler_error();
void snda_ecs_release_handler_error(SNDAECSHandlerError* error);
```

用户创建和销毁相应数据结构时，都必须使用本 SDK 提供的相应方法。

### 5.1.2 接口定义

盛大云存储 C 语言 SDK 相应接口都具有下面的格式：

```
SNDAECSErrorCode
snda_ecs_{method}(
SNDAECSHandler* handler,
const char* accesskey,
const char* secretkey,
… …,
SNDAECSResult* ret);
```

**返回值：SNDAECSErrorCode**

只有在该接口调用成功获得服务端响应（正确响应或者错误相应）时，返回 SNDA_ECS_SUCCESS，其他情况下返回其他值。

- 当返回 SNDA_ECS_SUCCESS 时，用户可以通过 ret->serverresponse->httpcode 来获得服务端返回的 http 状态码，并判断服务端是否成功响应响应请求。
  a) 当服务端给出错误响应时，用户可以调用
     SNDAECSErrorResponseContent*`snda_ecs_to_error_response`(SNDAECSResult*)
     来获取服务端详细的错误响应信息。
  b) 当服务端给出正确响应时，对于没有消息体的请求，操作结束；对于有消息体的请求，用户可以调用 `snda_ecs_to_xxx`(SNDAECSResult*)来获得相应的消息结构。具体在接口介绍中会详细介绍。
- 当返回非 SNDA_ECS_SUCCESS 时，用户可以通过 ret->error->handlererrmsg 获取一个可读的客户端错误信息

参数说明：

**输入参数：SNDAECSHandler* handler**

盛大云存储 C 语言 SDK HTTP 请求句柄，用户需要通过 **snda_ecs_init_handler()**获取该句柄指针，该句柄可以在单线程中多次重复使用，在确认不在使用后，通过调用 **snda_ecs_release_handler(SNDAECSHandler*)**释放其占用资源。

**输入参数：const char* accesskey**

用户在盛大云存储申请的用来标识用户身份的标识。具体参见盛大云存储开发者文档。

**输入参数：const char* secretkey**

盛大云存储颁发给用户的密钥，和 accesskey 一一对应。

**输出参数：SNDAECSResult* ret**

盛大云存储 C 语言 SDK 通用输出结构，其结构体为：

```
typedef struct SNDAECSResult {
        SNDAECSHandlerError* error;
SNDAECSServerResponse * serverresponse;
} SNDAECSResult;
```

其中 SNDAECSHandlerError* error 在接口返回非 SNDA_ECS_SUCCESS 时，包含可读的错误信息。

SNDAECSServerResponse * serverresponse 在接口返回 SNDA_ECS_SUCCESS 时，包含相应的响应消息。

用户必须通过 snda_ecs_init_result()来获取该结构体指针，并且在不再需要使用时调用 snda_ecs_release_result()来释放相应资源。

当用户希望在下一个调用中重用该结构体之前，必须调用 snda_ecs_reset_result(SNDAECSResult*)来重新初始化该结构体。

### 5.1.3 环境初始化

本 SDK 存在两个全局环境初始化和相应清理的函数。用户在使用本 SDK 之前，必须调用并且只能调用一次相应接口，它们是：

```
snda_ecs_global_init();          // init before any sdk action once and only once
snda_ecs_global_uninit();        // clear after any sdk action once and only once
```

## 5.2 Service 相关接口

### 5.2.1 获取所有 Bucket

该接口对应盛大云存储开发者文档中的 GetService，用户可以通过该操作列出所有 Bucket 信息。可通过设置参数 ssl 控制是否使用 SSL 安全加密操作。

接口定义：

```
/**
* get service
* @param SNDAECSHandler* handler, the handler you had initialized by
invoking snda_ecs_init_handler()
```

```
* @param const char* accesskey,your accessKey
* @param const char* secretkey,your secretKey
* @param int ssl,whether to use https
* @param SNDAECSResult* ret,SNDAECSResult* created from
snda_ecs_init_result(), if you want to reuse this pointer, MAKE SURE
invoke snda_ecs_reset_result(SNDAECSResult*) to reset this pointer
to initial status.
* return SNDAECSErrorCode
*/
SNDAECSErrorCode snda_ecs_get_service(
                 SNDAECSHandler* handler,
                 const char* accesskey,
                 const char* secretkey,
                 int ssl,
                 SNDAECSResult* ret);
```

详细使用实例：

```
void get_service_example(const char* accesskey, const char* secretkey,int ssl) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode ;
    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_get_service(handler, accesskey,secretkey, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf ("ClientErrorMessage:%s\n", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode == 200) {
        SNDAECSGetServiceResultContent* content =
                snda_ecs_to_get_service_result(ret);
        // show get service content
        for (; content; content = content->next) {
            printf ("BucketName:%s\n", content->bucketname);
            printf ("CreationTime:%s\n", content->creationtime);
        }
        // ALWAYS REMEMBER to release pointer resource by releated release method
        snda_ecs_release_get_service_result_content(content);
    } else {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        printf ("ErrorCode:%s\n", content->code);
        printf ("ErrorMessage:%s\n", content->message);
        printf ("Resource:%s\n", content->resource);
        printf ("RequestId:%s\n", content->requestid);
```

```
        printf ("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
    }
    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

## 5.3 Bucket 相关接口

### 5.3.1 创建 Bucket

该接口对应于盛大云存储 API 中的 PUT Bucket 接口，该接口可以创建一个新的 Bucket。
接口定义：

```
/**
* put bucket
* @param SNDAECSHandler* handler, the handler you had initialized by
invoking snda_ecs_init_handler()
* @param const char* accesskey,your accessKey
* @param const char* secretkey,your secretKey
* @param const char* bucketname,your bucketname
* @param const char* region,region of your bucket,region currently
 support "huadong-1", "huabei-1"
* @param int ssl,whether to use https
* @param SNDAECSResult* ret,SNDAECSResult* created from
snda_ecs_init_result(), if you want to reuse this pointer, MAKE SURE
invoke snda_ecs_reset_result(SNDAECSResult*)
*              to reset this pointer to initial status.
* return SNDAECSErrorCode
*/
SNDAECSErrorCode snda_ecs_put_bucket(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* region,
                            int ssl,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void put_bucket_example( const char* accesskey,const char* secretkey,const char* bucketname,const char*
region, int ssl) {
```

```
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;
    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    // when put bucket successfully, no value returned
    retcode = snda_ecs_put_bucket(handler, accesskey, secretkey, bucketname, region, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300){
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        printf ("ErrorCode:%s\n", content->code);
        printf ("ErrorMessage:%s\n", content->message);
        printf ("Resource:%s\n", content->resource);
        printf ("RequestId:%s\n", content->requestid);
        printf ("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
    } else {
        printf ("Put bucket success and the http code is:%d\n", ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.3.2 列出指定 Bucket 下所有 Objects

该接口对应于盛大云存储 API 的 Get Bucket 接口，通过该接口可以获得指定 Bucket 中的 Object 信息列表，请求时可以通过一些查询条件来限制返回的结果。

接口定义：

```
/**
 * Get bucket
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* prefix,the prefix parameter to the
 *       key of the object you want to retrieve
 * @param const char* marker,the key to start with
 * @param const char* delimiter,the character your use to
```

```
 *       group keys
 * @param int maxkeys,the maximum number of keys returned
 *       in the response body
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
* @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *       follow any "Location: " header that the server
 *       sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *       redirections to follow. Use this option alongside
 *       followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *       snda_ecs_init_result(), if you want to reuse this
 *       pointer, MAKE SURE invoke snda_ecs_reset_result
 *       (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_get_bucket(
                  SNDAECSHandler* handler,
                  const char* accesskey,
                  const char* secretkey,
                  const char* bucketname,
                  const char* prefix,
                  const char* marker,
                  const char* delimiter,
                  int maxkeys,
                  const char* region,
                  int ssl,
                  SNDAECSFollowLocation followlocation,
                  long maxredirects,
                  SNDAECSResult* ret);
```

详细使用实例：

```
void get_bucket_example(const char* accesskey, const char* secretkey,
        const char* bucketname, const char* prefix, const char* marker,
        const char* delimiter, int maxkeys, const char* region, int ssl,
        SNDAECSFollowLocation followlocation, long maxredirects) {


    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;
    SNDAECSGetBucketResultContent* content = 0;
```

```
snda_ecs_global_init();
handler = snda_ecs_init_handler();
ret = snda_ecs_init_result();
retcode = snda_ecs_get_bucket(handler, accesskey,
        secretkey, bucketname, prefix, marker, delimiter, maxkeys, region,
        ssl, followlocation, maxredirects, ret);
if (retcode != SNDA_ECS_SUCCESS) {
    printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
} else if (ret->serverresponse->httpcode == 200) {
    content = snda_ecs_to_get_bucket_result(ret);
    if (content) {
        if (content->bucketname) {
            printf("bucket:%s\n", content->bucketname);
        }
        if (content->prefix) {
            printf("prefix:%s\n", content->prefix);
        }
        if (content->marker) {
            printf("marker:%s\n", content->marker);
        }
        if (content->delimiter) {
            printf("delimiter:%s\n", content->delimiter);
        }
        if (content->nextmarker) {
            printf("nextmarker:%s\n", content->nextmarker);
        }
        printf("maxkeys:%d\n", content->maxkeys);
        printf("istruncated:%s\n", content->istruncated ? "true" : "false");

        printf("CONTENTS/\n");
        if (content->objects) {
            SNDAECSObjectContent* object = content->objects;
            while (object) {
                printf("\tCONTENT/\n");
                printf("\t\tobjectname:%s\n", object->objectname);
                printf("\t\tlastmodified:%s\n", object->lastmodified);
                printf("\t\tetag:%s\n", object->etag);
                printf("\t\tsize:%ld\n", object->size);
                object = object->next;
                printf("\t/CONTENT\n");
            }
        }
        printf("/CONTENTS\n");
```

```
                printf("COMMONPREFIXES/\n");
                if (content->commonprefixes) {
                    SNDAECSCommonPrefix* object = content->commonprefixes;
                    while (object) {
                        printf("\tCOMMONPREFIX/\n");
                        printf("\t\tcommonprefix:%s\n", object->commonprefix);
                        object = object->next;
                        printf("\t/COMMONPREFIX\n");
                    }
                }
                printf("/COMMONPREFIXES\n");
            }
            snda_ecs_release_get_bucket_result_content(content);
        } else if (ret->serverresponse->httpcode >= 300) {
            SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
            printf("ErrorCode:%s\n", content->code);
            printf("ErrorMessage:%s\n", content->message);
            printf("Resource:%s\n", content->resource);
            printf("RequestId:%s\n", content->requestid);
            printf("AllErrorMessage:%s\n", content->fullbody);
            snda_ecs_release_error_response_content(content);
        } else {
            printf("The http code is:%d\n", ret->serverresponse->httpcode);
        }

        snda_ecs_release_handler(handler);
        snda_ecs_release_result(ret);
}
```

### 5.3.3 删除指定 Bucket

该接口对应于盛大云存储 API 中的 DELETE Bucket 接口，可通过该接口删除指定的 Bucket，注意：必须确保要删除的 Bucket 中没有任何数据。

接口定义：

```
/**
 * Delete bucket
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* region,region of your bucket,region
```

```
 *        currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *        follow any "Location: " header that the server
 *        sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *        redirections to follow. Use this option alongside
 *        followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_delete_bucket(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void delete_bucket_example(const char* accesskey, const char* secretkey,
        const char* bucketname, const char* region, int ssl,
        SNDAECSFollowLocation followlocation, long maxredirects) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;
    SNDAECSErrorResponseContent* content = 0;
    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_delete_bucket(handler, accesskey,
            secretkey, bucketname, region, ssl, followlocation, maxredirects,
            ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
```

```
    } else if (ret->serverresponse->httpcode >= 300) {
        content = snda_ecs_to_error_response(ret);
        printf("HttpCode:%d\n",ret->serverresponse->httpcode);
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
        printf("RequestId:%s\n", content->requestid);
        printf("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Delete bucket success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.3.4 设置 Bucket Policy

该接口对应于盛大云存储 API 中的 PUT Bucket Policy 接口，该接口通过 policy 子资源来增加或替换指定的 Bucket 的 Policy。如果该 Bucket 已经存在了 Policy,那么该操作会替换原有的 Policy。

接口定义：

```
/**
 * Put bucket policy
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* policy,your bucket policy
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_put_bucket_policy(
                            SNDAECSHandler* handler,
                            const char* accesskey,
```

```
                                const char* secretkey,
                                const char* bucketname,
                                const char* policy,
                                int ssl,
                                SNDAECSResult* ret);
```

详细使用实例：

```
void put_bucket_policy_example(const char* accesskey, const char* secretkey,
        const char* bucketname, const char * policy, int ssl) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode ;
    SNDAECSErrorResponseContent* content = 0;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_put_bucket_policy(handler, accesskey,
            secretkey, bucketname, policy, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        content = snda_ecs_to_error_response(ret);
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
        printf("RequestId:%s\n", content->requestid);
        printf("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Put bucket policy success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.3.5 获取 Bucket Policy

该接口对应于盛大云存储 API 的 GET Bucket Policy 接口，该接口用于获取指定 Bucket 的 Policy。

接口定义：

```
/**
 * Get bucket policy
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_get_bucket_policy(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            int ssl,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void get_bucket_policy_example(const char* accesskey, const char* secretkey,
        const char* bucketname, int ssl) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;
    SNDAECSErrorResponseContent* content = 0;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_get_bucket_policy(handler, accesskey,
            secretkey, bucketname, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        content = snda_ecs_to_error_response(ret);
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
```

```
            printf("RequestId:%s\n", content->requestid);
            printf("AllErrorMessage:%s\n", content->fullbody);
            snda_ecs_release_error_response_content(content);
        } else if (ret->serverresponse->httpcode == 200) {

            char* policy = (char*)malloc(ret->serverresponse->responsebody->retbodysize + 1);
            policy[ret->serverresponse->responsebody->retbodysize] = '\0';

            memcpy(policy,(char*)(ret->serverresponse->responsebody->retbody),
            ret->serverresponse->responsebody->retbodysize
            );
            printf ("bucket:%s\n", bucketname);
            printf ("policy:%s\n", policy);
            snda_ecs_free_char_ptr(policy);

        } else {
            printf("Get bucket policy success and the http code is:%d\n",
                    ret->serverresponse->httpcode);
        }

        snda_ecs_release_handler(handler);
        snda_ecs_release_result(ret);
}
```

### 5.3.6 删除 Bucket Policy

该接口对应于盛大云存储 API 中的 Delete Bucket Policy，用户可以通过该操作删除指定的 Bucket 的 policy。

接口定义：

```
/**
 * Delete bucket policy
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
```

```
SNDAECSErrorCode snda_ecs_delete_bucket_policy(
                                SNDAECSHandler* handler,
                                const char* accesskey,
                                const char* secretkey,
                                const char* bucketname,
                                int ssl,
                                SNDAECSResult* ret);
```

详细使用实例：

```
void delete_bucket_policy_example(const char* accesskey, const char* secretkey,
        const char* bucketname, int ssl) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;
    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_delete_bucket_policy(handler,
            accesskey, secretkey, bucketname, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {

        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        printf("HTTP Code:%d\n",ret->serverresponse->httpcode);
        printf("hello");
        if(content){
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
        printf("RequestId:%s\n", content->requestid);
        printf("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
        }
    } else {
        printf("Delete bucket policy success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.3.7 获取 Bucket Location

该接口对应于 API 中的 GET Bucket Location,通过该请求可以获取目标 Bucket 所在的区域（Region）信息

接口定义：

```
/**
 * Get bucket location
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_get_bucket_location(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            int ssl,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void get_bucket_location_example(const char* accesskey, const char* secretkey,
        const char* bucketname, int ssl) {
    SNDAECSHandler* handler;
    SNDAECSResult* ret;
    SNDAECSErrorCode retcode ;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_get_bucket_location(handler, accesskey,
            secretkey, bucketname, ssl, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
```

```
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if(content){
          printf("ErrorCode:%s\n", content->code);
          printf("ErrorMessage:%s\n", content->message);
          printf("Resource:%s\n", content->resource);
          printf("RequestId:%s\n", content->requestid);
          printf("AllErrorMessage:%s\n", content->fullbody);
          snda_ecs_release_error_response_content(content);
        }
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
    } else if (ret->serverresponse->httpcode == 200) {
        SNDAECSBucketLocation* location = snda_ecs_to_bucket_location(ret);
        printf("bucket:%s\n", bucketname);
        printf("location:%s\n", location->location);
        snda_ecs_release_bucket_location(location);
    } else {
        printf("Get bucket location success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

## 5.3.8 列出指定 Bucket 下的所有未完成的 Multipart Upload

该接口对应于盛大云存储 API 的 List Multipart Upload 接口，通过该接口可列出指定 Bucket 下的所有未完成的 Multipart Upload。可通过设置查询字符来限制返回结果。

接口的定义：

```
/**
 * List_multipart_uploads
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* prefix,the prefix parameter to the
 *       key of the multipart upload you want to retrieve
 * @param const char* keymarker,the key to start with
```

```
 * @param const char* uploadidmarker,the uploadid to start with
 * @param const char* delimiter,the character your use to
 *       group keys
 * @param int maxuploads,the maximum number of keys returned
 *       in the response body
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *       follow any "Location: " header that the server
 *       sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *       redirections to follow. Use this option alongside
 *       followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *       snda_ecs_init_result(), if you want to reuse this
 *       pointer, MAKE SURE invoke snda_ecs_reset_result
 *       (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_list_multipart_uploads(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* prefix,
                            const char* keymarker,
                            const char* uploadidmarker,
                            const char* delimiter,
                            int maxuploads,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret)
```

详细使用实例：

```
void list_multipart_uploads_example(const char* accesskey,
        const char* secretkey, const char* bucketname, const char* prefix,
        const char* keymarker, const char * uploadidmarker,
        const char* delimiter, int maxuploads, const char* region, int ssl,
        SNDAECSFollowLocation followlocation, long maxredirects) {
```

```
SNDAECSHandler* handler = 0;
SNDAECSResult* ret = 0;
SNDAECSErrorCode retcode;

snda_ecs_global_init();
handler = snda_ecs_init_handler();
ret = snda_ecs_init_result();

retcode = snda_ecs_list_multipart_uploads(handler,
        accesskey, secretkey, bucketname, prefix, keymarker,
        uploadidmarker, delimiter, maxuploads, region, ssl, followlocation,
        maxredirects, ret);
if (retcode != SNDA_ECS_SUCCESS) {
    printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
} else if (ret->serverresponse->httpcode < 300) {
    SNDAECSMultipartUploadsContent* content =
            snda_ecs_to_multipart_uploads_content(ret);
    if (content) {
        SNDAECSMultipartUpload* upload = 0;
        SNDAECSCommonPrefix* object = 0;
        printf("Bucket:%s\n", content->bucket);
        printf("Prefix:%s\n", content->prefix);
        printf("Delimiter:%s\n", content->delimiter);
        printf("KeyMarker:%s\n", content->keymarker);
        printf("UploadIdMarker:%s\n", content->uploadidmarker);
        printf("NextKeyMarker:%s\n", content->nextkeymarker);
        printf("NextUploadIdMarker:%s\n", content->nextuploadidmarker);
        printf("IsTruncated:%d\n", content->istruncated);
        printf("MaxUploads:%d\n", content->maxuploads);

        printf("UPLOADS/\n");
        upload = content->upload;
        while (upload) {
            printf("\tUPLOAD/\n");
            printf("\t\tKey:%s\n", upload->key);
            printf("\t\tUploadId:%s\n", upload->uploadid);
            printf("\t\tInitiatedTime:%s\n", upload->initiatedtime);
            upload = upload->next;
            printf("\t/UPLOAD\n");
        }
        printf("/UPLOADS\n");

        printf("COMMONPREFIXES/\n");
```

```
                object = content->commonprefixes;
                while (object) {
                        printf("\tCOMMONPREFIX/\n");
                        printf("\t\tPrefix:%s\n", object->commonprefix);
                        object = object->next;
                        printf("\tCOMMONPREFIX/\n");
                }
                printf("/COMMONPREFIXES\n");
        }
        snda_ecs_release_multipart_uploads_content(content);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if (content) {
                if (content->code) {
                        printf("ErrorCode:%s\n", content->code);
                }
                if (content->message) {
                        printf("ErrorMessage:%s\n", content->message);
                }
                if (content->resource) {
                        printf("Resource:%s\n", content->resource);
                }
                if (content->requestid) {
                        printf("RequestId:%s\n", content->requestid);
                }
                if (content->fullbody) {
                        printf("AllErrorMessage:%s\n", content->fullbody);
                }
        }
        snda_ecs_release_error_response_content(content);
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
    } else {
        printf("The http code is:%d\n", ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

## 5.4 Object 相关接口

### 5.4.1 新建 Object

该接口对应于盛大云存储 API 中的 PUT OBJECT 接口，该接口用来上传一个新的 Object 到指定的 Bucket 中，数据的最大长度限制为 5GB。

接口定义：

```
/**
 * Put Object
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param CallbackFunPtr readFun, used as CURLOPT_READDATA,usually
is snda_ecs_put_object_body
 * @param void* inputstream, data stream for upload ,usually a pointer
of file opened with "rb"
 * @param long contentlength,the size of the object,in bytes
 * @param const SNDAECSUserObjectMeta* userobjectmeta,used in request
headers
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_put_object(
                        SNDAECSHandler* handler,
                        const char* accesskey,
                        const char* secretkey,
                        const char* bucketname,
                        const char* objectname,
                        CallbackFunPtr readFun,
                        void* inputstream,
                        long contentlength,
                        const SNDAECSUserObjectMeta*
                        userobjectmeta,
```

```
                                const char* region, int ssl,
                                SNDAECSResult* ret)
```

详细使用实例：

```
void put_object_example(const char* accesskey, const char* secretkey,
        const char* bucketname, const char *region, const char * localfile,
        const char *objectname, int ssl) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSUserObjectMeta* objectmeta = 0;
    FILE* fd = 0;
    long flength ;
    SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    objectmeta = snda_ecs_init_user_object_meta();
    snda_ecs_set_object_type(objectmeta, "binary/octet-stream");
    // furthermore, user can set user metas with snda_ecs_add_object_user_metas()
    // all key of user metas must begin with "x-snda-meta-", and case insensitive
    snda_ecs_add_object_user_metas(objectmeta, "x-snda-meta-1",
            "this is my user meta 1");
    snda_ecs_add_object_user_metas(objectmeta, "x-SNDA-metA-2",
            "WOO, the seconde user meta");

    fd = fopen(localfile, "rb");
    if(!fd) {
        printf("Please check your file!\n");
        return ;
    }
    fseek(fd, 0L, SEEK_END);
    flength = ftell(fd);
    fseek(fd, 0, 0);
    retcode = snda_ecs_put_object(handler, accesskey,
            secretkey, bucketname, objectname, snda_ecs_put_object_body, fd,
            flength, objectmeta, region, ssl, ret);
    snda_ecs_release_user_object_meta(objectmeta);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
```

```
            if(content) {
            printf("ErrorCode:%s\n", content->code);
            printf("ErrorMessage:%s\n", content->message);
            printf("Resource:%s\n", content->resource);
            printf("RequestId:%s\n", content->requestid);
            printf("AllErrorMessage:%s\n", content->fullbody);
            snda_ecs_release_error_response_content(content);
            }
            if(ret->serverresponse->httpcode == 505) {
               printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
            }
      } else {
            printf("Put Object success and the http code is:%d\n",
                        ret->serverresponse->httpcode);
      }

      snda_ecs_release_handler(handler);
      snda_ecs_release_result(ret);
}
```

## 5.4.2 获取 Object 的 meta 信息

该接口对应于盛大云存储 API 的 HEAD Object 接口，通过该接口可以获取指定 Object 的元数据信息。
接口定义：

```
/**
 * Head Object
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param SNDAECSByteRange* byterange,the specified range bytes of
the object.
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *        follow any "Location: " header that the server
 *        sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
```

```
 *        redirections to follow. Use this option alongside
 *        followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_head_object(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            SNDAECSByteRange* byterange,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void head_object_example(const char* accesskey, const char* secretkey,
        const char* bucketname, const char *region, const char * objectname,
        long byterangefirst, long byterangelast, int ssl, int followlocation,
        int maxredirects) {
    SNDAECSHandler* handler = snda_ecs_init_handler();
    SNDAECSResult* ret = snda_ecs_init_result();
    SNDAECSByteRange* byterangeptr = 0;
    SNDAECSErrorCode retcode ;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    byterangeptr = snda_ecs_init_byte_range();
    byterangeptr->first = byterangefirst;
    byterangeptr->last = byterangelast;

    retcode = snda_ecs_head_object(handler, accesskey,
            secretkey, bucketname, objectname, byterangeptr, region, ssl,
            followlocation, maxredirects, ret);
    snda_ecs_release_byte_range(byterangeptr);
    if (retcode != SNDA_ECS_SUCCESS) {
```

```
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if(content) {
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
        printf("RequestId:%s\n", content->requestid);
        printf("AllErrorMessage:%s\n", content->fullbody);
        snda_ecs_release_error_response_content(content);
        }
        if(ret->serverresponse->httpcode == 505) {
          printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
    } else {
        SNDAECSObjectMeta* objectmeta = snda_ecs_to_object_meta(ret);
        SNDAECSKVList* p = 0;
        printf("Etag:%s\n", objectmeta->etag);
        printf("Content-Type:%s\n", objectmeta->contenttype);
        printf("Content-Length:%s\n", objectmeta->lastmodified);
        printf("Last-Modified:%s\n", objectmeta->lastmodified);
        p = objectmeta->usermetas;
        for (; p; p = p->next) {
            printf("p->key:%s\n", p->value);
        }

        snda_ecs_release_object_meta(objectmeta);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.4.3 获取（下载）Object

该接口对应盛大云存储 API 的 GET OBJECT 接口，可通过该接口获取指定 Object 内容。
接口定义：

```
/**
 * Get Object
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
```

```
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param SNDAECSByteRange* byterange,the specified range bytes of
the object.
 * @param CallbackFunPtr writeFun,used as
CURLOPT_READFUNCTION,usually is snda_ecs_write_fun();
 * @param  void* outputstream, usually a pointer of file opend with
"wb";
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *        follow any "Location: " header that the server
 *        sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *        redirections to follow. Use this option alongside
 *        followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_get_object(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            SNDAECSByteRange* byterange,
                            CallbackFunPtr writeFun,
                            void* outputstream,
                            const char* region, int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret)
```

详细使用实例：

```
void get_object_example(const char* accesskey, const char* secretkey,
        const char* bucket, const char *region, const char * objectname,
        const char * locafile, long byterangefirst, long byterangelast,
```

```
        int ssl, int followlocation, int maxredirects) {

SNDAECSHandler* handler = 0;
SNDAECSResult* ret = 0;
SNDAECSByteRange* byterangeptr = 0;
FILE* writefd = 0;
SNDAECSErrorCode retcode;

snda_ecs_global_init();
handler = snda_ecs_init_handler();
ret = snda_ecs_init_result();
byterangeptr = snda_ecs_init_byte_range();

byterangeptr->first = byterangefirst;
byterangeptr->last = byterangelast;

writefd = fopen(locafile, "wb");
if(!writefd) {
    printf("Please check your localfile path!\n");
    return;
}
retcode = snda_ecs_get_object(handler, accesskey,
            secretkey, bucket, objectname, byterangeptr, snda_ecs_write_fun,
            writefd, region, ssl, followlocation, maxredirects, ret);
fclose(writefd);
snda_ecs_release_byte_range(byterangeptr);
if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
} else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        printf("Get Object failed and the http code is:%d\n",
                ret->serverresponse->httpcode);

        if (content) {
            if (content->code) {
                printf("ErrorCode:%s\n", content->code);
            }
            if (content->message) {
                printf("ErrorMessage:%s\n", content->message);
            }
            if (content->resource) {
                printf("Resource:%s\n", content->resource);
            }
```

```
                if (content->requestid) {
                        printf("RequestId:%s\n", content->requestid);
                }
                if (content->fullbody) {
                        printf("AllErrorMessage:%s\n", content->fullbody);
                }
        }
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Get Object success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.4.4 删除 Object

该接口对应于盛大云存储 API 的 DELETE Object 接口，用于删除指定的 Object
接口定义：

```
/**
 * Delete Object
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *        follow any "Location: " header that the server
 *        sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *        redirections to follow. Use this option alongside
 *        followlocation.
```

```
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_delete_object(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void delete_object_example(const char* accesskey, const char* secretkey,
        const char* bucket, const char *region, const char * objectname,
        int ssl, int followlocation, int maxredirects) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;

    SNDAECSErrorCode retcode ;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
   retcode = snda_ecs_delete_object(handler, accesskey,
            secretkey, bucket, objectname, region, ssl, followlocation,
            maxredirects, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if (content) {
            if (content->code) {
                printf("ErrorCode:%s\n", content->code);
            }
            if (content->message) {
```

```
                printf("ErrorMessage:%s\n", content->message);
            }
            if (content->resource) {
                printf("Resource:%s\n", content->resource);
            }
            if (content->requestid) {
                printf("RequestId:%s\n", content->requestid);
            }
            if (content->fullbody) {
                printf("AllErrorMessage:%s\n", content->fullbody);
            }
        }
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Delete Object success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

## 5.4.5 拷贝 Object

该接口对应盛大云存储开发者文档中的 Copy Object，用户可以通过该操作将已经存在于存储上的 Object 拷贝至指定 Buket 下。

接口定义：

```
 /**
  * @param SNDAECSHandler* handler, the handler you had
  *       initialized by invoking snda_ecs_init_handler()
  * @param const char* accesskey,your accessKey
  * @param const char* secretkey,your secretKey
  * @param const char* destbucketname,the name of the destination
bucket
  * @param const char* destobjectname,the key of the destination
object
  * @param const char* srcbucketname,the name of the source bucket
  * @param const char* srcobjectname,the key of the source object
  * @param const SNDAECSUserObjectMeta* userobjectmeta,used in
request headers
  * @param const char* region,region of your bucket,region
  *       currently support "huadong-1", "huabei-1"
  * @param int ssl,whether to use https
```

```
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_copy_object(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* destbucketname,
                            const char* destobjectname,
                            const char *srcbucketname,
                            const char * srcobjectname,
                            const SNDAECSUserObjectMeta*
                           userobjectmeta,
                            const char* region, int ssl,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void copy_object_example(const char* accesskey, const char* secretkey,
        const char* destbucketname, const char * destobjectname,
        const char * srcbucketname, const char * srcobjectname,
        const char *region, int ssl) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSUserObjectMeta* objectmeta = 0;
    SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    objectmeta = snda_ecs_init_user_object_meta();
    snda_ecs_set_object_type(objectmeta, "binary/octet-stream");
    // furthermore, user can set user metas with snda_ecs_add_object_user_metas()
    // all key of user metas must begin with "x-snda-meta-", and case insensitive
    snda_ecs_add_object_user_metas(objectmeta, "x-snda-meta-1",
            "this is my user meta 1");
    snda_ecs_add_object_user_metas(objectmeta, "x-SNDA-metA-2",
            "WOO, the seconde user meta");

    retcode = snda_ecs_copy_object(handler, accesskey,
```

```
                secretkey, destbucketname, destobjectname, srcbucketname,
                srcobjectname, objectmeta, region, ssl, ret);
    snda_ecs_release_user_object_meta(objectmeta);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if(content) {
        printf("ErrorCode:%s\n", content->code);
        printf("ErrorMessage:%s\n", content->message);
        printf("Resource:%s\n", content->resource);
        printf("RequestId:%s\n", content->requestid);
        printf("AllErrorMessage:%s\n", content->fullbody);
        }
        if(ret->serverresponse->httpcode == 505) {
          printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Copy Object success and the http code is:%d\n",
                ret->serverresponse->httpcode);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

## 5.4.6 初始化一个 Multipart Upload

该接口对应于盛大云存储 API 的 Initiate Multipart Upload，用于初始化一个 Multipart Upload。

接口定义：

```
/**
 * Initiate Multipart upload
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param const char* region,region of your bucket,region
 *        currently support "huadong-1", "huabei-1"
 * @param const SNDAECSUserObjectMeta* userobjectmeta,used
```

```
 *        in request headers
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *        follow any "Location: " header that the server
 *        sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *        redirections to follow. Use this option alongside
 *        followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_initiate_multipart_upload(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            const SNDAECSUserObjectMeta*
                            userobjectmeta,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret)
```

详细使用实例：

```
void initiate_multipart_upload_example( const char* accesskey,const char* secretkey,const char* bucket,
                                        const char *region,const char * objectname,int ssl,
                                        int followlocation,int maxredirects){


    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSUserObjectMeta* objectmeta = 0;
    char contenttype[S_SNDA_ECS_CONTENT_TYPE_LEN];
    SNDAECSErrorCode retcode ;


    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
```

```
    objectmeta = snda_ecs_init_user_object_meta();

    snda_ecs_set_object_type(objectmeta, snda_ecs_get_content_type(objectname, contenttype));
    retcode = snda_ecs_initiate_multipart_upload(handler, accesskey, secretkey,
                                bucket, objectname,objectmeta, region, ssl, followlocation, maxredirects,ret);
    snda_ecs_release_user_object_meta(objectmeta);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300){
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if(content) {
            if(content->code) {
            printf ("ErrorCode:%s\n", content->code);
            }
            if(content->message) {
            printf ("ErrorMessage:%s\n", content->message);
            }
            if(content->resource) {
            printf ("Resource:%s\n", content->resource);
            }
            if(content->requestid) {
            printf ("RequestId:%s\n", content->requestid);
            }
            if(content->fullbody) {
            printf ("AllErrorMessage:%s\n", content->fullbody);
            }
        }
          if(ret->serverresponse->httpcode == 505) {
             printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
          }
        snda_ecs_release_error_response_content(content);
    } else {
        SNDAECSInitiateMultipartUploadResult*  content  =
snda_ecs_to_initiate_multipart_upload_result(ret);
        printf ("Initiate multipart upload success and the http code is:%d\n", ret->serverresponse->httpcode);
         if(content) {
            if(content->bucket) {
            printf ("Bucket:%s\n", content->bucket);
            }
            if(content->key) {
            printf ("Key:%s\n", content->key);
            }
            if(content->uploadid) {
```

```
            printf ("UploadId:%s\n", content->uploadid);
            }
        }
        snda_ecs_release_initiate_multipart_upload_result(content);
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.4.7 终止一个 Multipart Upload

该接口对应于盛大云存储 API 中的 Abort Multipart Upload 接口，可通过该接口终止一个指定的 Multipart Upload。当一个 Multipart Upload 被终止后，其 UploadId 也一同作废，且该 Multipart Upload 中的所有 Part 所占用的存储空间均会被释放。

接口定义：

```
/**
 * Abort_multipart_uploads
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param const char* uploadid,your uploadid for
 *        the multipart upload
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *       follow any "Location: " header that the server
 *       sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *       redirections to follow. Use this option alongside
 *       followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *       snda_ecs_init_result(), if you want to reuse this
 *       pointer, MAKE SURE invoke snda_ecs_reset_result
 *       (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
```

```
 */
SNDAECSErrorCode snda_ecs_abort_multipart_upload(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            const char* uploadid,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret)
```

详细使用实例：

```
void abort_multipart_upload_example(const char* accesskey,
        const char* secretkey, const char* bucket, const char *region,
        const char * objectname, const char * uploadid, int ssl,
        int followlocation, int maxredirects) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode ;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();

    retcode = snda_ecs_abort_multipart_upload(handler,
            accesskey, secretkey, bucket, objectname, uploadid, region, ssl,
            followlocation, maxredirects, ret);

    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if (content) {
            if (content->code) {
                printf("ErrorCode:%s\n", content->code);
            }
            if (content->message) {
                printf("ErrorMessage:%s\n", content->message);
            }
```

```
                if (content->resource) {
                        printf("Resource:%s\n", content->resource);
                }
                if (content->requestid) {
                        printf("RequestId:%s\n", content->requestid);
                }
                if (content->fullbody) {
                        printf("AllErrorMessage:%s\n", content->fullbody);
                }
        }
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey,uploadid!\n");
        }
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Abort multipart upload success and the http code is %d\n",
                ret->serverresponse->httpcode);
    }
    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

### 5.4.8 上传一个 Part

该接口对应盛大云存储开发者文档中的 Upload Part，用户可以通过该操作上传一个 Part 到指定的 Multipart Upload 中。

接口定义：

```
/**
 * Upload Part
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,your object name
 * @param const char* uploadid,your uploadid for multipart upload
 * @param int partnumber,partnumber of this part
 * @param CallbackFunPtr readFun, used as CURLOPT_READDATA,usually
is snda_ecs_put_object_body
 * @param void* inputstream, data stream for upload ,usually a pointer
of file opened with "rb"
 * @param long contentlength,the size of the object,in bytes
 * @param const char* contentmd5,contentmd5 of this part(can be null)
```

```
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_upload_part(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            const char* uploadid,
                            int partnumber,
                            CallbackFunPtr readFun,
                            void* inputstream,
                            long contentlength,
                            const char* contentmd5,
                            const char* region,
                            int ssl,
                            SNDAECSResult* ret);
```

详细使用实例：

```
void upload_part_example(const char* accesskey, const char* secretkey,
            const char* bucket, const char *region, const char * objectname,
            const char * uploadid, const char * localfile, int ssl, int partnumber) {
    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    FILE* fd = 0;
    long filelength;
    char * contentmd5 = 0;
    SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    fd = fopen(localfile, "rb");
    if(!fd) {
        printf("please check your localfile path!\n");
```

```
    return ;
}
fseek(fd, 0L, SEEK_END);
filelength = ftell(fd);
fseek(fd, 0, 0);

retcode = snda_ecs_upload_part(handler, accesskey,
            secretkey, bucket, objectname, uploadid, partnumber,
            snda_ecs_put_object_body, fd, filelength, contentmd5, region, ssl,
            ret);

if (retcode != SNDA_ECS_SUCCESS) {
    printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
} else if (ret->serverresponse->httpcode >= 300) {
    SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
    if (content) {
        if (content->code) {
            printf("ErrorCode:%s\n", content->code);
        }
        if (content->message) {
            printf("ErrorMessage:%s\n", content->message);
        }
        if (content->resource) {
            printf("Resource:%s\n", content->resource);
        }
        if (content->requestid) {
            printf("RequestId:%s\n", content->requestid);
        }
        if (content->fullbody) {
            printf("AllErrorMessage:%s\n", content->fullbody);
        }
    }

    if(ret->serverresponse->httpcode == 505) {
       printf("Please check your bucketname,accessKey,SecretAccessKey,uploadid!\n");
    }
    snda_ecs_release_error_response_content(content);
} else {
    printf("Upload part success and the http code is %d\n",
            ret->serverresponse->httpcode);
}
snda_ecs_release_handler(handler);
snda_ecs_release_result(ret);
```

```
}
```

### 5.4.9 拷贝上传一个 Part

该接口对应于盛大云存储 API 中的 Upload Part – copy,通过该接口可完成将一个云存储中已存在的 Object 拷贝给指定的 Multipart upload part 的一个 part,可通过对应参数的设置修改该 part 的属性。

接口定义:

```
/**
 * Upload Part - Copy
 * @param SNDAECSHandler* handler, the handler you had
 *        initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,the name of the destination bucket
 * @param const char* objectname,the key of the destination object
 * @param const char* uploadid,your uploadid for multipart upload
 * @param int partnumber,partnumber of this part
 * @param const char * region,your destination bucket's region
 * @param const SNDAECSUserObjectMeta* userobjectmeta,used in
request headers
 * @param const char* srcbucketname,the name of the source bucket
 * @param const char* srcobjectname,the key of the source object
 * @param int ssl,whether to use https
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *        snda_ecs_init_result(), if you want to reuse this
 *        pointer, MAKE SURE invoke snda_ecs_reset_result
 *        (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
 SNDAECSErrorCode snda_ecs_upload_part_copy(
                    SNDAECSHandler* handler,
                    const char* accesskey,
                    const char* secretkey,
                    const char* bucketname,
                    const char* objectname,
                    const char* uploadid,
                    int partnumber,
                    const char * region,
                    const SNDAECSUserObjectMeta* userobjectmeta,
                    const char* sbucket,
                    const char* sobjectname,
                    int ssl,
```

```
                                SNDAECSResult* ret);
```

详细使用实例：

```
void multipart_upload_copy_example(const char* accesskey, const char* secretkey,
        const char* destbucketname, const char * destobjectname,const char * uploadid,
        const int partnumber,const char * srcbucketname, const char * srcobjectname,
        const char *region, int ssl) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSUserObjectMeta* objectmeta = 0;
    SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    objectmeta = snda_ecs_init_user_object_meta();
    // furthermore, user can set user metas with snda_ecs_add_object_user_metas()
    // all key of user metas must begin with "x-snda-meta-", and case insensitive
    //snda_ecs_add_object_user_metas(objectmeta, "x-SNDA-metA-2",
    //        "WOO, the seconde user meta");

    retcode = snda_ecs_upload_part_copy(handler,accesskey,secretkey, destbucketname,destobjectname,
        uploadid,partnumber,region,objectmeta,srcbucketname,srcobjectname, ssl,ret);
    snda_ecs_release_user_object_meta(objectmeta);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        printf("Error,HTTP code %d\n",ret->serverresponse->httpcode );
        if(content) {
            printf("ErrorCode:%s\n", content->code);
            printf("ErrorMessage:%s\n", content->message);
            printf("Resource:%s\n", content->resource);
            printf("RequestId:%s\n", content->requestid);
            printf("AllErrorMessage:%s\n", content->fullbody);
        }
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey!\n");
        }
        snda_ecs_release_error_response_content(content);
```

```
    } else {
        char * result = snda_ecs_get_full_response(ret);

        printf("Upload Part - copy success and the http code is:%d\n",
                ret->serverresponse->httpcode);
        if(result){
          printf("Response:\n%s\n",result);
        }
    }

    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
    snda_ecs_global_uninit();
}
```

## 5.4.10 列出已上传的 Parts

该接口对应盛大云存储开发者文档中的 List Parts，用户可以通过该操作列出一个 Multipart Upload 已上传的 Part。

接口定义：

```
/**
 * List_Parts
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,the object name of the
 *        multipart upload
 * @param const char* uploadid,the uploadid of the multipart
 *        upload
 * @param const char* partnumbermarker,the part to start with
 * @param int maxparts,the maximum number of parts returned
 *       in the response body
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *       follow any "Location: " header that the server
 *       sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *       redirections to follow. Use this option alongside
```

```
 *       followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *       snda_ecs_init_result(), if you want to reuse this
 *       pointer, MAKE SURE invoke snda_ecs_reset_result
 *       (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
SNDAECSErrorCode snda_ecs_list_parts(
                            SNDAECSHandler* handler,
                            const char* accesskey,
                            const char* secretkey,
                            const char* bucketname,
                            const char* objectname,
                            const char* uploadid,
                            int partnumbermarker,
                            int maxparts,
                            const char* region,
                            int ssl,
                            SNDAECSFollowLocation followlocation,
                            long maxredirects,
                            SNDAECSResult* ret)
```

详细使用实例：

```
void list_parts_example(const char* accesskey, const char* secretkey,
        const char* bucket, const char *region, const char * objectname,
        const char * uploadid, int ssl, int followlocation,
        int partnumbermarker, int maxparts, int maxdirects) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    retcode = snda_ecs_list_parts(handler, accesskey,
            secretkey, bucket, objectname, uploadid, partnumbermarker,
            maxparts, region, ssl, followlocation, maxdirects, ret);
    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode < 300) {
        SNDAECSMultipartsContent* content = snda_ecs_to_multipart_parts(ret);
```

```
        SNDAECSMultipartsPart* part = 0;
        if (content) {
            printf("Bucket:%s\n", content->bucket);
            printf("Key:%s\n", content->key);
            printf("UploadId:%s\n", content->uploadid);
            printf("MaxParts:%d\n", content->maxparts);
            printf("IsTruncated:%d\n", content->istruncated);
            printf("PartNumberMarker:%d\n", content->partnumbermarker);
            printf("NextPartNumberMarker:%d\n", content->nextpartnumbermarker);

            printf("PARTS/\n");
            part = content->parts;
            while (part) {
                printf("\tPART/\n");
                printf("\t\tPartNumber:%d\n", part->partnumber);
                printf("\t\tSize:%ld\n", part->size);
                printf("\t\tLastModified:%s\n", part->lastmodified);
                printf("\t\tETag:%s\n", part->etag);
                part = part->next;
                printf("\t/PART\n");
            }
            printf("/PARTS\n");
        }
        snda_ecs_release_multiparts_content(content);
    } else {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if (content) {
            if (content->code) {
                printf("ErrorCode:%s\n", content->code);
            }
            if (content->message) {
                printf("ErrorMessage:%s\n", content->message);
            }
            if (content->resource) {
                printf("Resource:%s\n", content->resource);
            }
            if (content->requestid) {
                printf("RequestId:%s\n", content->requestid);
            }
            if (content->fullbody) {
                printf("AllErrorMessage:%s\n", content->fullbody);
            }
        }
```

```
            if(ret->serverresponse->httpcode == 505) {
               printf("Please check your bucketname,accessKey,SecretAccessKey,uploadid!\n");
            }
            snda_ecs_release_error_response_content(content);
        }
        snda_ecs_release_handler(handler);
        snda_ecs_release_result(ret);
}
```

## 5.4.11 完成一个 Multipart Upload

该接口对应盛大云存储开发者文档中的 Complete Multipart Upload，用户可以通过该操作来完成 Multipart Upload,合并其包含的所有 Part,并在云存储中产生一个新的 Objcet.

接口定义：

```
/**
 * Complete multipart upload
 * @param SNDAECSHandler* handler, the handler you had
 *       initialized by invoking snda_ecs_init_handler()
 * @param const char* accesskey,your accessKey
 * @param const char* secretkey,your secretKey
 * @param const char* bucketname,your bucketname
 * @param const char* objectname,the object name of the
 *        multipart upload
 * @param const char* uploadid,the uploadid of the multipart
 *        upload
 * @param const SNDAECSMultipartsMeta* partsmeta,partsmetas of
 *        the multipart upload
 * @param const char* region,region of your bucket,region
 *       currently support "huadong-1", "huabei-1"
 * @param int ssl,whether to use https
 * @param SNDAECSFollowLocation followlocation,whether to
 *       follow any "Location: " header that the server
 *       sends as part of the HTTP header
 * @param long maxredirects,the maximum amount of HTTP
 *       redirections to follow. Use this option alongside
 *       followlocation.
 * @param SNDAECSResult* ret,SNDAECSResult* created from
 *       snda_ecs_init_result(), if you want to reuse this
 *       pointer, MAKE SURE invoke snda_ecs_reset_result
 *       (SNDAECSResult*) to reset this pointer to initial status.
 * return SNDAECSErrorCode
 */
```

```
SNDAECSErrorCode snda_ecs_complete_multipart_upload(
                                SNDAECSHandler* handler,
                                const char* accesskey,
                                const char* secretkey,
                                const char* bucketname,
                                const char* objectname,
                                const char* uploadid,
                                const SNDAECSMultipartsMeta* partsmeta,
                                const char* region,
                                int ssl,
                                SNDAECSFollowLocation followlocation,
                                long maxredirects,
                                SNDAECSResult* ret)
```

详细使用实例：

```
void complete_multipart_upload_example(const char* accesskey,
        const char* secretkey, const char* bucket, const char *region,
        const char * objectname, const char * uploadid, int ssl,
        int followlocation, int maxdirects) {

    SNDAECSHandler* handler = 0;
    SNDAECSResult* ret = 0;
    SNDAECSMultipartsMeta* metas = 0;
    SNDAECSMultipartsMeta* p = 0;
   SNDAECSErrorCode retcode;

    snda_ecs_global_init();
    handler = snda_ecs_init_handler();
    ret = snda_ecs_init_result();
    metas = snda_ecs_init_multiparts_meta();
    p = metas;
    p->partnumber = 1;
    snda_ecs_copy_string(&(p->etag), "\"58fda622140205b3d6a2457415d301f2\"");
    p->next = snda_ecs_init_multiparts_meta();
    p = p->next;
    p->partnumber = 2;
    snda_ecs_copy_string(&(p->etag), "\"58fda622140205b3d6a2457415d301f2\"");
    p->next = snda_ecs_init_multiparts_meta();
    p = p->next;
    p->partnumber = 3;
    snda_ecs_copy_string(&(p->etag), "\"296e2bd0ce7124b6fbda05873c261dfb\"");
```

```
    retcode = snda_ecs_complete_multipart_upload(handler,
                accesskey, secretkey, bucket, objectname, uploadid, metas, region,
                ssl, followlocation, maxdirects, ret);

    snda_ecs_release_multiparts_meta(metas);

    if (retcode != SNDA_ECS_SUCCESS) {
        printf("ClientErrorMessage:%s", ret->error->handlererrmsg);
    } else if (ret->serverresponse->httpcode >= 300) {
        SNDAECSErrorResponseContent* content = snda_ecs_to_error_response(ret);
        if (content) {
            if (content->code) {
                printf("ErrorCode:%s\n", content->code);
            }
            if (content->message) {
                printf("ErrorMessage:%s\n", content->message);
            }
            if (content->resource) {
                printf("Resource:%s\n", content->resource);
            }
            if (content->requestid) {
                printf("RequestId:%s\n", content->requestid);
            }
            if (content->fullbody) {
                printf("AllErrorMessage:%s\n", content->fullbody);
            }
        }
        if(ret->serverresponse->httpcode == 505) {
           printf("Please check your bucketname,accessKey,SecretAccessKey,uploadid!\n");
        }
        snda_ecs_release_error_response_content(content);
    } else {
        printf("Complete multipart upload success and the http code is %d\n",
                ret->serverresponse->httpcode);
    }
    snda_ecs_release_handler(handler);
    snda_ecs_release_result(ret);
}
```

如 果 在 使 用 中 遇 到 任 何 问 题 ， 请 在

**http://forum.grandcloud.cn/ 反馈，我们将及时跟进。谢谢！**