盛大云存储 C# 语言SDK使用说明

目录

1. 系统要求

使用盛大云存储的C# SDK 需要Microsoft .NET framework 2.0或以上版本支持。

2. 盛大云存储的基本概念

2.1AccessKey

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）。

2.2 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。

2.3 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。

2.4 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

2.5 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。

2.6 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

3. 基本数据类型

3.1 GrandCloud.CS.Model.CSBucket

盛大云存储的Bucket对象

3.2 GrandCloud.CS.Model.CSObject

盛大云存储的Object对象

4. 存储服务对象

4.1 GrandCloud.CS.GrandCloudCS

云存储服务基本类型，抽象类，主要负责控制与存储服务的交互，提供了当前存储服务的全部服务API。

4.2 GrandCloud.CS.GrandCloudCSClient

云存储服务API的实现类，可通过GrandCloud.CS. CSClientFactory.

CreateGrandCloudCSClient()产生GrandCloudCS对象或者自行构造此类。 构造GrandCloudCSClient时需传入认证信息。认证信息可通过在构造时传入accesskey和sceretkey或者在App.config提供，示例如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
            <add key="CSAccessKey" value=""/>
            <add key="CSSecretKey" value=""/>
    </appSettings>
</configuration>
```

4.3 GrandCloud.CS.GrandCloudCSConfig

云存储服务对象配置服务类。可通过此类设置service url，user agent，proxy，通讯协议，出现错误时的最大重试次数，是否使用SecureString保存认证信息等信息。

5. 身份验证对象

5.1 CSCredentials

SNDA身份验证对象。

5.2 BasicCSCredentials

CSCredentials的子类对象，对应直接提供accesskey和secretkey构造认证对象的实现。可通过在构造时提供useSecureString参数为true来保存密钥为C#的SecureString对象。

5.3 EnvironmentCSCredential

对应从App.config中读取accesskey和secretkey构造认证对象的实现。

6. 使用说明

云存储C# SDK 的对象操作支持同步操作与异步操作。如创建bucket的同步操作方法为CreateBucket()，则异步操作为BeginCreateBucket()及EndCreateBucket()，以此类推。
存储对象操作以构建Request，返回Response的方式调用。

6.1 创建云存储服务对象

```
GrandCloudCSConfig config = new GrandCloudCSConfig()
                .WithUserAgent("test-agent")
                .WithUseSecureStringForGrandCloudSecretKey(true);
GrandCloudCS client
 = GrandCloud.CS.CSClientFactory.CreateGrandCloudCSClient(config);
```

6.2获得用户Bucket列表

```
ListBucketsResponse response = client.ListAllBuckets();
int i = 0;
foreach (CSBucket bucket in response.Buckets)
{
    i++;
    System.Console.WriteLine("Bucket {0}: name = {1}, location =
{2}, create date = {3}", i, bucket.BucketName,
bucket.BucketRegionName, bucket.CreationDate);
}
```

6.3创建Bucket

```
PutBucketRequest request = new PutBucketRequest();
request.BucketName = bucketName;
request.BucketRegion = bucketRegion;
client.CreateBucket(request);
```

6.4 获取Bucket是否存在

```
bool exists = GrandCloudCSUtil. DoesCSBucketExist(“test-bucket”,
client);
```

6.5 删除Bucket

```
DeleteBucketRequest req = new DeleteBucketRequest();
req.BucketName = bucketName;
client.DeleteBucket(req);
```

6.6 上传Object

```
// simple object put
PutObjectRequest request = new PutObjectRequest();
request.WithContentBody("this is a test")
```

```
                    .WithBucketName(bucketName)
                    .WithKey(keyName);

CSResponse response = client.PutObject(request);
response.Dispose();

// put a more complex object with some metadata and http headers.
PutObjectRequest titledRequest = new PutObjectRequest();
titledRequest.WithMetaData("title", "the title")
                    .WithContentBody("this object has a title")
                    .WithBucketName(bucketName)
                    .WithKey(keyName);

CSResponse responseWithMetadata = client.PutObject(titledRequest);
responseWithMetadata.Dispose();

// put a file.
PutObjectRequest fileRequest = new PutObjectRequest();
fileRequest.WithFilePath("c:\\test.gif")
                    .WithBucketName(bucketName)
                    .WithKey(keyName)
                    .WithGenerateChecksum(true);
CSResponse responseWithMetadata = client.PutObject(fileRequest);
responseWithMetadata.Dispose();
```

6.7 以REDUCED_REDUNDENCY方式上传Object

```
PutObjectRequest request = new PutObjectRequest();
request.WithContentBody("this is a test")
                    .WithBucketName(bucketName)
                    .WithKey(keyName)
                    .WithStorageClass(CSStorageClass.
ReducedRedundancy);
```

6.8 获取Object的Metadata

```
    GetObjectMetadataRequest request = new
GetObjectMetadataRequest() .WithBucketName(bucketName).WithKey(obj
ectKey);
    using (GetObjectMetadataResponse response =
client.HeadObject(request))
   {
      WebHeaderCollection headers = response.Headers;
```

```
    foreach (string key in headers.Keys)
    {
      Console.WriteLine("Response Header: {0}, Value: {1}", key,
headers.Get(key));
    }
  }
```

6.9下载Object

```
GetObjectRequest request = new
GetObjectRequest().WithBucketName(bucketName).WithKey(keyNam
e);
using (GetObjectResponse response = client.GetObject(request))
{
  string title = response.Metadata["x-snda-meta-title"];
  Console.WriteLine("The object's title is {0}", title);
  string dest =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.
Desktop), keyName);
  if (!File.Exists(dest))
  {
     response.WriteResponseStreamToFile(dest);
  }
}
```

6.10 删除Object

```
DeleteObjectRequest request = new DeleteObjectRequest();
request.WithBucketName(bucketName)
                    .WithKey(keyName);
using (DeleteObjectResponse response = client.DeleteObject(request))
{
  WebHeaderCollection headers = response.Headers;
  foreach (string key in headers.Keys)
  {
    Console.WriteLine("Response Header: {0}, Value: {1}", key,
headers.Get(key));
  }
}
```

6.11 获取Bucket location

```
GetBucketLocationRequest request = new GetBucketLocationRequest()
                    .WithBucketName(bucketName);
```

```
using (GetBucketLocationResponse response =
client.GetBucketLocation(request))
{
    Console.WriteLine("bucket: {0}, location: {1}", bucketName,
response.Location);
}
```

6.12 获取Bucket下Object列表

```
ListObjectsRequest request = new ListObjectsRequest();
request.BucketName = bucketName;
using (ListObjectsResponse response = client.ListObjects(request))
{
    foreach (CSObject entry in response.CSObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.Key,
entry.Size);
    }
}
Console.WriteLine("----------------------");
// list only things starting with "foo"
request.WithPrefix("foo");
using (ListObjectsResponse response = client.ListObjects(request))
{
    foreach (CSObject entry in response.CSObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.Key,
entry.Size);
    }
}
Console.WriteLine("----------------------");
// list only things that come after "bar" alphabetically
request.WithPrefix(null)
    .WithMarker("bar");
using (ListObjectsResponse response = client.ListObjects(request))
{
    foreach (CSObject entry in response.CSObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.Key,
entry.Size);
    }
}
Console.WriteLine("----------------------");
```

```
// only list 3 things
request.WithPrefix(null)
    .WithMarker(null)
    .WithMaxKeys(3);
using (ListObjectsResponse response = client.ListObjects(request))
{
    foreach (CSObject entry in response.CSObjects)
    {
        Console.WriteLine("key = {0} size = {1}", entry.Key,
entry.Size);
    }
}
```

6.13 获取Bucket policy

```
GetBucketPolicyRequest request = new GetBucketPolicyRequest()
    .WithBucketName(bucketName);
using (GetBucketPolicyResponse response =
client.GetBucketPolicy(request))
{
    Console.WriteLine(response.Policy);
}
```

6.14 设置Bucket policy

```
PutBucketPolicyRequest request = new PutBucketPolicyRequest()
                .WithBucketName(bucketName)
                .WithPolicy(policy);
PutBucketPolicyResponse response = client.SetBucketPolicy(request);
```

6.15 删除Bucket policy

```
DeleteBucketPolicyRequest request = new PutBucketPolicyRequest()
                .WithBucketName(bucketName);

DeleteBucketPolicyResponse response =
client.DeleteBucketPolicy(request);
```

6.16 指定开始一个MultiUpload

```
InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest()
                .WithBucketName(bucketName)
```

```
                .WithKey(keyName);
InitiateMultipartUploadResponse response =
client.InitiateMultipartUpload(initRequest);
return response.UploadId;
```

6.17 上传一个MultiUpload对象的part

```
FileStream file = File.OpenRead(fileName);
long filesize = file.Length;
long parts =  filesize / (5 * 1024 * 1024);
file.Close();
if (parts == 0)
{
    Console.WriteLine("The file is not bigger than 5M.");
    return;
}
int part = 1;
long partsize = 5 * 1024 * 1024;
List<string> etags = new List<string>();
while (filesize > 0)
{
    partsize = filesize > partsize ? 5 * 1024 * 1024 : filesize;
    UploadPartRequest upReq = new UploadPartRequest()
    .WithBucketName(bucketName)
    .WithKey(keyName)
    .WithGenerateChecksum(true)
    .WithUploadId(uploadId)
    .WithPartNumber(part)
    .WithPartSize(partsize)
    .WithFilePath(fileName)
    .WithFilePosition(5 * 1024 * 1024 * (part - 1));
    UploadPartResponse response = client.UploadPart(upReq);
    etags.Add(response.ETag);
    part++;
    filesize -= partsize;
}
```

6.18 列出未完成的part

```
ListPartsRequest request = new ListPartsRequest()
        .WithBucketName(bucketName)
        .WithKey(keyName)
        .WithUploadId(uploadId);
```

```
ListPartsResponse response = client.ListParts(request);
List<PartETag> etags = new List<PartETag>();
foreach (PartDetail detail in response.Parts)
{
  etags.Add(new PartETag(detail.PartNumber, detail.ETag));
}
```

6.19 列出Bucket下所有MultiUpload对象

```
ListMultipartUploadsRequest request = new
ListMultipartUploadsRequest()
        .WithBucketName(bucketName);
ListMultipartUploadsResponse response =
client.ListMultipartUploads(request);
```

6.20 完成MultiUpload操作

```
CompleteMultipartUploadRequest request = new
CompleteMultipartUploadRequest()
        .WithBucketName(bucketName)
        .WithKey(keyName)
        .WithUploadId(uploadId)
        .WithPartETags(etags);
CompleteMultipartUploadResponse response =
client.CompleteMultipartUpload(request);
```

6.21 取消MultiUpload操作

```
AbortMultipartUploadRequest request = new
AbortMultipartUploadRequest()
        .WithBucketName(bucketName)
        .WithKey(keyName)
        .WithUploadId(uploadId);
            AbortMultipartUploadResponse response =
client.AbortMultipartUpload(request);
```

6.22 创建带签名的URL

```
GetPreSignedUrlRequest request = new GetPreSignedUrlRequest();
request.BucketName = bucketName;
request.Expires = new DateTime(2019, 12, 31);
request.Verb = HttpVerb.GET;
request.Protocol = Protocol.HTTP;
string url = csClient.GetPreSignedURL(request);
```

7. 异常

7.1 GrandCloud.CS.GrandCloudCSException

在访问云存储过程中，所有没有能够正常完成服务请求的操作，都会返回GrandCloudCSException，由Exception派生而来，ServiceException的对象中，保存了以下HTTP请求失败返回的全部信息：response code，response status，response date，response body，response header，以及这次request的信息：request method， request path， request host；并对错误返回的response body进行了解析，并会得出以下由存储服务器获得到的错误返回的响应：error code， error message， error request id， error resource。

7.2 GrandCloud.CS.GrandCloudServiceException

当使用FallbackCredentialsFactory. GetCredentials()方法返回CSCredentials 时，若无法找到密钥，将返回此异常。

8. 代码注释

更详细的API说明请参见代码注释。

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！