数据保险箱

API 文档

腾讯云

文档目录

API 文档
更新历史

最近更新时间：2021-11-24 10:42:48

第1次发布

发布时间：2020-06-05 15:12:29

本次发布包含了以下内容：

新增接口：

- GetService
- PutCoffer
- GetCoffer
- DeleteCoffer
- PutCofferPolicy
- GetCofferPolicy
- DeleteCofferPolicy
- PutCofferLifecycle
- GetCofferLifecycle
- PutObject
- GetObject
- HeadObject
- InitiateMultipartUpload
- UploadPart
- ListParts
- CompleteMultipartUpload
- AbortMultipartUpload
- ListMultipartUploads

简介

最近更新时间：2021-11-24 10:43:10

欢迎使用腾讯云数据保险箱（Cloud Data Coffer Service，CDCS）。

数据保险箱是腾讯云为用户提供的更高安全系数的数据加密存储服务，用于满足客户安全且精细化的数据安全管理需求，符合监管和合规要求。

本文档将说明如何使用 API 快速接入数据保险箱，通过数据保险箱可以进行大批量文件的上传、下载和多地域备份管理。

术语信息

本文档的主要术语如下：

名称	描述
AppID	开发者访问数据保险箱时，拥有的用户维度唯一资源标识，用以标识资源。
SecretID	开发者拥有的项目身份识别 ID，用以身份认证。
SecretKey	开发者拥有的项目身份密钥。
Coffer	数据保险箱中用于存储数据的容器。
Object	数据保险箱中存储的具体文件，是存储的基本实体。

API 概览

最近更新时间：2021-11-24 10:43:31

保险箱管理

接口名称	接口功能
PutCoffer	创建保险箱
DeleteCoffer	删除保险箱
GetService	查询保险箱列表

文件管理

接口名称	接口功能
GetCoffer	获取文件列表
GetObject	下载文件
HeadObject	获取文件信息
PutObject	上传文件

生命周期管理

接口名称	接口功能
PutCofferLifecycle	创建生命周期
GetCofferLifecycle	获取生命周期配置

访问策略

接口名称	接口功能
GetCofferPolicy	查看访问策略
PutCofferPolicy	设置访问策略

接口名称	接口功能
DeleteCofferPolicy	清除访问策略

分片上传

接口名称	接口功能
InitiateMultipartUpload	初始化分片上传
UploadPart	将对象按照分块的方式上传到保险箱
ListParts	查询特定分块上传中已上传的块
CompleteMultipartUpload	完成整个分块上传
AbortMultipartUpload	舍弃一个分块上传并删除已上传的块
ListMultipartUploads	查询正在进行中的分块上传任务

调用方式
请求签名

最近更新时间：2023-02-03 11:03:30

使用数据保险箱服务时，可通过 RESTful API 对数据保险箱服务发起 HTTP 签名请求，数据保险箱服务的服务器端将会对请求发起者的身份进行验证。

数据保险箱服务基于密钥 HMAC（Hash Message Authentication Code）的自定义方案进行身份验证。

可对 API 请求进行多方面的安全防护：

- **请求者身份验证**：通过访问者唯一 ID 和密钥确定请求者身份。
- **防止传输数据篡改**：对数据签名并检验，保障传输内容完整性。
- **防止签名被盗用**：对签名设置时效，避免签名盗用并重复使用。

准备工作

1. 在访问管理控制台的 API 密钥管理 页面中获取 AppID、SecretId 和 SecretKey。
2. 确定开发语言：支持但不限于 Java、PHP、C#、C++、Node.js、Python，根据不同的开发语言，确定对应的 HMAC-SHA1、SHA1 和 UrlEncode 函数。其中，HMAC-SHA1 和 SHA1 函数以 UTF-8 编码字符串为输入，以16进制小写字符串为输出。UrlEncode 基于 UTF-8 编码，包括 ASCII 范围内的可打印字符，下列特殊符号也应被编码：

字符	十进制	十六进制	字符	十进制	十六进制
(空格)	32	20	;	59	3B
!	33	21	<	60	3C
"	34	22	=	61	3D
#	35	23	>	62	3E
$	36	24	?	63	3F
%	37	25	@	64	40
&	38	26	[	91	5B
'	39	27	\	92	5C
(	40	28	]	93	5D

字符	十进制	十六进制	字符	十进制	十六进制
)	41	29	^	94	5E
*	42	2A	`	96	60
+	43	2B	{	123	7B
,	44	2C	|	124	7C
/	47	2F	}	125	7D
:	58	3A	---	---	---

⚠ 注意：

- 建议用户使用子账号密钥 + 环境变量的方式调用 SDK，提高 SDK 使用的安全性。为子账号授权时，请遵循 最小权限指引原则，防止泄漏目标存储桶或对象之外的资源。
- 如果您一定要使用永久密钥，建议遵循 最小权限指引原则 对永久密钥的权限范围进行限制。

签名步骤

步骤1：生成 KeyTime

1. 获取当前时间对应的 Unix 时间戳 StartTimestamp，Unix 时间戳是从 UTC（协调世界时，或 GMT 格林威治时间）1970年1月1日0时0分0秒（北京时间 1970年1月1日8时0分0秒）起至现在的总秒数。
2. 根据上述时间戳和期望的签名有效时长算出签名过期时间对应的 Unix 时间戳 EndTimestamp。
3. 拼接签名有效时间，格式为StartTimestamp;EndTimestamp，即为 KeyTime。

示例： 1557902800;1557910000

步骤2：生成 SignKey

使用 HMAC-SHA1 以 SecretKey 为密钥，以 KeyTime 为消息，计算消息摘要（哈希值），即为 SignKey。

示例： 36bcd76dbb8c9f066472fec403df8a34cab34c77

步骤3：生成 UrlParamList 和 HttpParameters

1. 遍历 HTTP 请求参数，生成 key 到 value 的映射 Map 及 key 的列表 KeyList，其中 key 转换为小写形式，value 使用 UrlEncode 编码，没有 value 的参数，则认为 value 为空字符串。例如请求路径为/?acl，则认为是/?acl=。

> **说明：**
> HTTP 请求参数，即请求路径中？以后的部分，例如请求路径为 /?versions%2F&delimiter=%2F&maxCount=10 ，则请求参数为 versions%2F&delimiter=%2F&maxCount=10 。

2. 将 KeyList 按照字典序排序。
3. 将 Map 和 KeyList 中的 key 使用 UrlEncode 编码，并再次转换为小写形式。
4. 按照 KeyList 的顺序拼接 Map 中的每一个键值对，格式为key1=value1&key2=value2&key3=value3，即为 HttpParameters。
5. 按照 KeyList 的顺序拼接 KeyList 中的每一项，格式为key1;key2;key3，即为 UrlParamList。

示例：

- 示例一：
  请求路径:/example-coffer/?delimiter=%2F&maxCount=10
  UrlParamList: delimiter;maxCount
  HttpParameters: delimiter=%2F&maxCount=10

> **注意：**
> 请求路径中的请求参数在实际发送请求时也会进行 UrlEncode，因此要注意不要重复执行 UrlEncode。

- 示例二：
  请求路径:/example-coffer?replications
  UrlParamList: replications
  HttpParameters: replications=

步骤4：生成 HeaderList 和 HttpHeaders

1. 遍历 HTTP 请求头部，生成 key 到 value 的映射 Map 及 key 的列表 KeyList，其中 key 转换为小写形式，value 使用 UrlEncode 编码。
2. 将 KeyList 按照字典序排序。
3. 将 Map 和 KeyList 中的 key 使用 UrlEncode 编码，并再次转换为小写形式。
4. 按照 KeyList 的顺序拼接 Map 中的每一个键值对，格式为key1=value1&key2=value2&key3=value3，即为 HttpHeaders。
5. 按照 KeyList 的顺序拼接 KeyList 中的每一项，格式为key1;key2;key3，即为 HeaderList。

示例：

请求头：

```
Host: cdcs.ap-shanghai.myqcloud.com
Date: Thu, 16 May 2019 03:15:06 GMT
Content-Type: application/json
Content-Length: 65535
```

计算得到 HeaderList 为 date;content-length;content-type;host **，HttpHeaders 为** date=Thu%2C%2016%20May%202019%2003%3A15%3A06%20GMT&content-length=65535&content-type=application%2Fjson&host=cdcs.ap-shanghai.myqcloud.com **。**

步骤5：生成 HttpString

根据 HTTP 方法、HTTP 请求路径、HttpParameters 和 HttpHeaders 生成 HttpString，格式为 HttpMethod\nUriPathname\nHttpParameters\nHttpHeaders\n **。**

其中：

- **HttpMethod 转换为小写，例如 get 或 put。**
- **UriPathname 为请求路径，例如/或/examplecoffer。**
- **\n为换行符。如果其中有字符串为空，前后的换行符需要保留，例如**get\n/examplecoffer\n\n\n**。**

步骤6：生成 StringToSign

根据 KeyTime 和 HttpString 生成 StringToSign，格式为 sha1\nKeyTime\nSHA1(HttpString)\n **。**

其中：

- **sha1 为固定字符串。**
- **\n为换行符。**
- **SHA1(HttpString) 为使用 SHA1 对 HttpString 计算的消息摘要。**

步骤7：生成 Signature

使用 HMAC-SHA1 以 SignKey 为密钥，以 StringToSign 为消息，计算消息摘要，即为 Signature。

步骤8：生成签名

根据 SecretId、KeyTime、HeaderList、UrlParamList 和 Signature 生成签名，格式为：

```
q-sign-algorithm=sha1&q-ak=SecretId&q-sign-time=KeyTime&q-key-time=KeyTime&q-header-list=HeaderList&q-url-param-list=UrlParamList&q-signature=Signature
```

签名使用

通过 RESTful API 对数据保险箱服务发起的 HTTP 签名请求，需要讲签名放在 HTTP Authorization 头，例如 Authorization: q-sign-algorithm=sha1&q-ak=...&q-sign-time=1557989753;1557996953&...&q-signature=...

> 说明：
> 上述示例中使用 ... 省略了部分具体签名内容。

代码示例

伪代码

```
KeyTime = [Now];[Expires]
SignKey = HMAC-SHA1([SecretKey], KeyTime)
HttpString = [HttpMethod]\n[HttpURI]\n[HttpParameters]\n[HttpHeaders]\n
StringToSign = sha1\nKeyTime\nSHA1(HttpString)\n
Signature = HMAC-SHA1(SignKey, StringToSign)
```

消息摘要算法示例

不同语言如何调用 HMAC-SHA1 可以参考下面的示例：

PHP

```php
$sha1HttpString = sha1('ExampleHttpString');

$signKey = hash_hmac('sha1', 'ExampleKeyTime', 'YourSecretKey');
```

Java

```java
import org.apache.commons.codec.digest.DigestUtils;
import org.apache.commons.codec.digest.HmacUtils;

String sha1HttpString = DigestUtils.sha1Hex("ExampleHttpString");

String signKey = HmacUtils.hmacSha1Hex("YourSecretKey", "ExampleKeyTime");
```

Python

```
import hmac
import hashlib

sha1_http_string = hashlib.sha1('ExampleHttpString'.encode('utf-8')).hexdigest()

sign_key = hmac.new('YourSecretKey'.encode('utf-8'), 'ExampleKeyTime'.encode('utf-8'), hashli
b.sha1).hexdigest()
```

Node.js

```
var crypto = require('crypto');

var sha1HttpString = crypto.createHash('sha1').update('ExampleHttpString').digest('hex');
var signKey = crypto.createHmac('sha1', 'YourSecretKey').update('ExampleKeyTime').digest('he
x');
```

Go

```
import (
"crypto/hmac"
"crypto/sha1"
)

h := sha1.New()
h.Write([]byte("ExampleHttpString"))
sha1HttpString := h.Sum(nil)

var hashFunc = sha1.New
h = hmac.New(hashFunc, []byte("YourSecretKey"))
h.Write([]byte("ExampleKeyTime"))
signKey := h.Sum(nil)
```

实际案例

准备工作

登录访问管理控制台的 API 密钥管理 页面获取其 AppID、SecretId 和 SecretKey，举例如下：

AppID	SecretId	SecretKey
1250000000	SecretId	SecretKey

上传对象

原始请求

```
PUT /example-coffer/example-file HTTP/1.1
Date: Thu, 16 May 2019 06:45:51 GMT
Host: cdcs.ap-beijing.myqcloud.com/example-coffer/example-file
Content-Type: text/plain
Content-Length: 13
Content-MD5: mQ/fVh815F3k6TAUm8m0eg==

ObjectContent
```

中间变量

- **KeyTime** = 1557989151;1557996351
- **SignKey** = eb2519b498b02ac213cb1f3d1a3d27a3b3c9bc5f
- **UrlParamList** = (empty string)
- **HttpParameters** = (empty string)
- **HeaderList** = content-length;content-md5;content-type;date;host;x-cdcs-acl;x-cdcs-grant-read
- **HttpHeaders** = content-length=13&content-md5=mQ%2FfVh815F3k6TAUm8m0eg%3D%3D&content-type=text%2Fplain&date=Thu%2C%2016%20May%202019%2006%3A45%3A51%20GMT&host=cdcs.ap-beijing.myqcloud.com
- **HttpString** = put\n/example-coffer/example-file\n\ncontent-length=13&content-md5=mQ%2FfVh815F3k6TAUm8m0eg%3D%3D&content-type=text%2Fplain&date=Thu%2C%2016%20May%202019%2006%3A45%3A51%20GMT&host=cdcs.ap-beijing.myqcloud.com\n
- **StringToSign** = sha1\n1557989151;1557996351\n8b2751e77f43a0995d6e9eb9477f4b685cca4172\n
- **Signature** = 3b8851a11a569213c17ba8fa7dcf2abec6935172

其中，(empty string) 代表长度为0的空字符串， \n 代表换行符。

签名后的请求

```
PUT /example-coffer/example-file HTTP/1.1
Date: Thu, 16 May 2019 06:45:51 GMT
Host: cdcs.ap-beijing.myqcloud.com
Content-Type: text/plain
Content-Length: 13
Content-MD5: mQ/fVh815F3k6TAUm8m0eg==
Authorization: q-sign-algorithm=sha1&q-ak=SecretId&q-sign-time=1557989151;1557996351&q-key-time=1557989151;1557996351&q-header-list=content-length;content-md5;content-type;date;host&q-url-param-list=&q-signature=3b8851a11a569213c17ba8fa7dcf2abec6935172

ObjectContent
```

公共请求头部

最近更新时间：2021-11-24 10:44:12

描述

本文档将为您介绍在使用 API 时会使用到的公共请求头部（Request Header），下文提到的请求头部在其他的具体 API 文档中不再赘述。

请求头部列表

Header 名称	是否必选	类型	描述
Authorization	是必选。	string	携带鉴权信息，用以验证请求合法性的签名信息。 针对公有读的对象可不携带此头部，如通过请求参数传递鉴权信息也无需携带此头部，详情请参见 请求签名 文档。
Content-Length	针对 PUT 和 POST 请求，此头部是必选项。	integer	RFC 2616中定义的 HTTP 请求内容长度（字节）。
Content-Type	针对有请求体的 PUT 和 POST 请求，此头部是必选项。	string	RFC 2616中定义的 HTTP 请求内容类型（MIME） 例如application/xml或image/jpeg。
Content-MD5	否。	string	RFC 1864中定义的请求体内容的16字节二进制 MD5 哈希值的 Base64 编码形式，最终的取值长度应为24个字符，请注意在编写代码时使用正确的方法和参数，例如ZzD3iDJdrMAAb00lgLLeig==。 此头部用于完整性检查，验证请求体在传输过程中是否发生变化。针对有请求体的 PUT 和 POST 请求（除 POST Object），强烈建议携带此头部。
Date	否。	string	RFC 1123中定义的 GMT 格式当前时间，例如Wed, 29 May 2019 04:10:12 GMT。

Header 名称	是否必选	类型	描述
Host	是。	string	请求的主机，形式为cdcs.<Region>.myqcloud.com。
x-cdcs-security-token	当使用临时密钥并通过Authorization携带鉴权信息时，此头部为必选项。	string	使用临时安全凭证时需要传入的安全令牌字段。

公共响应头部

最近更新时间：2021-11-24 10:50:40

描述

本文档将为您介绍在使用 API 时会出现的公共响应头部（Response Header），下文提到的响应头部在其他具体 API 文档中不再赘述。

响应头部列表

Header 名称	类型	描述
Content-Length	string	RFC 2616中定义的 HTTP 响应内容长度（字节）。
Content-Type	string	RFC 2616中定义的 HTTP 响应内容类型（MIME）。
Connection	Enum	RFC 2616中定义，表明响应完成后是否会关闭网络连接。枚举值：keep-alive，close。
Date	string	RFC 1123中定义的 GMT 格式服务端响应时间，例如Wed, 29 May 2019 04:10:12 GMT。
ETag	string	ETag 全称为 Entity Tag，是对象被创建时标识对象内容的信息标签，可用于检查对象的内容是否发生变化，例如"8e0b617ca298a564c3331da28dcb50df"。此头部并不一定返回对象的 MD5 值，而是根据对象上传和加密方式而有所不同。
Last-Modified	string	对象的最近一次上传的时间，例如Fri, 10 Apr 2020 18:17:25 GMT。
Server	string	接收请求并返回响应的服务器的名称，默认值：tencent-cdcs。
Transfer-Encoding	string	RFC 2616中定义的传输编码格式。
x-cdcs-request-id	string	每次请求发送时，服务端将会自动为请求生成一个 ID。
x-cdcs-trace-id	string	每次请求出错时，服务端将会自动为这个错误生成一个 ID，仅当请求出错时才会在响应中包含此头部。

访问策略语言概述

最近更新时间：2020-06-09 11:38:57

> **注意：**
> 在给子用户或者协作者添加访问策略时，请务必根据业务需要，按照最小权限原则进行授权。如果您直接授予子用户或者协作者所有资源 (resource:*)，或者所有操作 (action:*) 权限，则存在由于权限范围过大导致数据安全风险。

概述

访问策略可用于授予访问数据保险箱资源的权限。访问策略使用基于 JSON 的访问策略语言。您可以通过访问策略语言授权指定委托人（principal）对指定的数据保险箱资源执行指定的操作。

访问策略语言描述了访问策略（Access Policy）的基本元素和用法。

访问策略中的元素

访问策略语言包含用于描述权限信息的一条或多条语句（statement），由以下部分组成：

- **委托人（principal）**：描述策略授权的实体，例如主账号、子账号、匿名用户等。
- **效力（effect）**：该元素是必填项，描述声明产生的结果是“允许”还是“显式拒绝”，包括 allow 和 deny 两种情况。
- **操作（action）**：该元素是必填项，描述允许或拒绝的操作。操作可以是 API（以 name 前缀描述）或者功能集（一组特定的 API，以 permid 前缀描述）。
- **资源（resource）**：描述授权的具体数据，资源是用六段式描述。
- **条件（condition）**：该元素是非必填项，描述策略生效的约束条件，条件包括操作符、操作键和操作值组成。条件值可包括 IP 地址等信息，有些服务允许您在条件中指定其他值。

元素用法

指定委托人

委托人 principal 元素用于指定被允许或拒绝访问资源的用户、账户、服务或其他实体。下面是指定 principal 的示例。

```
"principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
```

```
]
}
```

授权主账户 UIN 100000000001权限：

```
"principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
}
```

授权子账户 UIN 100000000011（主账户 UIN 为100000000001）权限：

> ⚠ 注意：
> 操作前需确保子账号已被添加到主账号的子账号列表中。

```
"principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000011"
]
}
```

指定效力

如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝（deny）对资源的访问，这样可确保用户无法访问该资源，即使有其他策略授予了访问权限的情况下也是如此。下面是指定允许效力的示例。

```
"effect" : "allow"
```

指定操作

数据保险箱定义了可在策略中指定的某一个特定的数据保险箱操作，指定的操作与发起的 API 请求操作完全一致。

保险箱操作

描述	对应的 API 接口

描述	对应的 API 接口
name/cdcs:GetService	GetService
name/cdcs:GetCoffer	GetCoffer（List Object）
name/cdcs:PutCoffer	PutCoffer
name/cdcs:DeleteCoffer	DeleteCoffer

对象操作

描述	对应的 API 接口
name/cdcs:GetObject	GetObject
name/cdcs:PutObject	PutObject
name/cdcs:CheckObject	HeadObject

生命周期管理

描述	对应的 API 接口
name/cdcs:PutCofferLifecycle	PutCofferLifecycle
name/cdcs:GetCofferLifecycle	GetCofferLifecycle

访问策略

描述	对应的 API 接口
name/cdcs:GetCofferPolicy	GetCofferPolicy
name/cdcs:PutCofferPolicy	PutCofferPolicy
name/cdcs:DeleteCofferPolicy	DeleteCofferPolicy

分片上传

描述	对应的 API 接口
name/cdcs:InitiateMultipartUpload	InitiateMultipartUpload
name/cdcs:UploadPart	UploadPart

描述	对应的 API 接口
name/cdcs:ListParts	ListParts
name/cdcs:CompleteMultipartUpload	CompleteMultipartUpload
name/cdcs:AbortMultipartUpload	AbortMultipartUpload
name/cdcs:ListMultipartUploads	ListMultipartUploads

指定允许操作的示例如下：

```
"action": [
"name/cdcs:GetObject",
"name/cdcs:CheckObject"
]
```

指定资源

资源（resource）元素描述一个或多个操作对象，如保险箱、对象等。所有资源均可采用下述的六段式描述方式。

```
qcs:project_id:service_type::account:resource
```

参数说明如下：

参数	是否必选	描述
qcs	是	qcloud service 的简称，表示是腾讯云的云服务。
project_id	可选	描述项目信息，仅为了兼容 CAM 早期逻辑。
service_type	是	描述产品简称，如 cdcs。
account	是	描述资源拥有者的主账号信息。目前支持两种方式描述的资源拥有者。 一种方式是 uin 方式，即主账号的 qq 号，表示为 uin/${OwnerUin}，如 uin/100000000001。 另外一种方式是 uid 方式，即主账号的 APPID，表示为 uid/${appid}，如 uid/1250000000。目前 数据保险箱 的资源拥有者统一使用 uid 的方式表述，即主账号的开发商 AppID。
resource	是	描述具体资源详情，在 数据保险箱 服务中使用 XML API 访问路径来描述。

下面是指定保险箱 examplecoffer-1250000000 的示例。

```
"resource": ["qcs::cdcs::uid/1250000000:examplecoffer-1250000000/*"]
```

下面是指定保险箱 examplecoffer-1250000000 中的 exampleobject 对象的示例。

```
"resource": ["qcs::cdcs::uid/1250000000:examplecoffer-1250000000/exampleobject"]
```

指定条件

访问策略语言可使您在授予权限时指定条件。例如限制用户访问来源，限制授权时间等。下面列出了目前支持的条件操作符列表以及通用的条件键和示例等信息。

条件操作符	含义	条件名	示例
ip_equal	IP 等于	qcs:ip	{"ip_equal":{"qcs:ip ":"10.121.2.0/24"}}
ip_not_equal	IP 不等于	qcs:ip	{"ip_not_equal":{"qcs:ip ":["10.121.1.0/24", "10.121.2.0/24"]}}
date_not_equal	时间不等于	qcs:current_time	{"date_not_equal":{"qcs:current_time":"2016-06-01T00:01:00Z"}}
date_greater_than	时间大于	qcs:current_time	{" date_greater_than ":{"qcs:current_time":"2016-06-01T00:01:00Z"}}
date_greater_than_equal	时间大于等于	qcs:current_time	{" date_greater_than_equal ":{"qcs:current_time":"2016-06-01T00:01:00Z"}}
date_less_than	时间小于	qcs:current_time	{" date_less_than ":{"qcs:current_time":"2016-06-01T 00:01:00Z"}}
date_less_than_equal	时间小于等于	qcs:current_time	{"date_less_than_equal ":{"qcs:current_time":"2016-06-01T00:01:00Z"}}

下面是满足来访 IP 为 10.121.2.0/24 网段内的示例。

```
"ip_equal":{"qcs:ip ":"10.121.2.0/24"}
```

下面是满足来访 IP 为 101.226.***.185 和 101.226.***.186 的示例。

```
"ip_equal": {
"qcs:ip": [
"101.226.***.185",
"101.226.***.186"
]
}
```

实际案例

允许主账号1234下的子账号5678，在访问来源 IP 为 101.226.***.185/101.226.***.186 时，对保险箱 examplecoffer-1250000000 中的对象执行 GET（下载）和 HEAD 操作。

```
{
"version": "2.0",
"statement": [
{
"principal": {
"qcs": [
"qcs::cam::uin/1234:uin/5678"
]
},
"action": [
"name/cdcs:GetObject",
"name/cdcs:CheckObject"
],
"condition": {
"ip_equal": {
"qcs:ip": [
"101.226.***.185",
"101.226.***.186"
]
}
```

```
},
"effect": "allow",
"resource": [
"qcs::cdcs::uid/1250000000:examplecoffer-1250000000/*"
]
}
]
}
```

地域和访问域名

最近更新时间：2020-06-09 10:18:57

简介

地域（Region）是腾讯云托管机房的分布地区，数据保险箱的数据存放在这些地域中。您可以通过数据保险箱，将数据进行多地域存储。通常情况下，数据保险箱建议您选择在与您业务最近的地域上创建，以满足低延迟、低成本以及合规性要求。

目前机房分布的地区如下：

地域	地域简称	域名
天津	ap-tianjin	cdcs.ap-tianjin.myqcloud.com

> 说明：
> 目前支持备份区域包括三个：天津、上海、广州。

内网和外网访问

腾讯云数据保险箱的访问域名使用了智能 DNS 解析，通过互联网在不同的运营商环境下，我们会检测并指向最优链路供您访问数据保险箱。

如果您在腾讯云内部署了服务用于访问数据保险箱，则同地域范围内访问将会自动被指向到内网地址，跨地域暂不支持内网访问，默认将会解析到外网地址。

保险箱管理
在指定账号下创建保险箱

最近更新时间：2023-10-08 09:17:18

接口描述

- **接口名称**：PutCoffer
- **接口功能**：该请求接　可以在指定账号下创建　个数据保险箱，创建数据保险箱的　户默认成为数据保险箱的持有者。

请求

请求示例

```
PUT /<CofferName-APPID> HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: GMT Date
Content-Length: 0
Authorization: Auth String
```

> **说明**：
> Authorization: Auth String 的详情，请参　请求签名　档。

请求参数

此接　　请求参数。

请求头

此接　仅使　公共请求头部，详情请参　公共请求头部　档。

请求体

该 API 接　请求的请求体具体节点内容为：

```
{
"Locations": {
```

```
"Location": ["ap-tianjin"]
},
"EncryptType": 0,
"KeyId": "",
"KeyRegion": ""
}
```

具体内容描述如下：

节点名称（关键字）	是否必选	类型	描述
EncryptType	否	int	加密类型： • 0 = 不加密（默认） • 1 = 使　公共主密钥 • 2 = 使　KMS 主密钥。
KeyId	否	string	主密钥的 ID，仅当 EncryptType = 2 时，填写该参数。
KeyRegion	否	string	主密钥的地域，仅当 EncryptType = 2 时，填写该参数。
Locations	是	Object	保险箱所在地域列表。
Location	是	Enum	保险箱所在地域，枚举值请参　地域和访问域名　档。 例如ap-beijing，ap-hongkong，eu-frankfurt等

响应

响应头

此接　仅返回公共响应头部，详情请参　公共响应头部　档。

响应体

此接　响应体为空。

错误码

此接　的特殊错误信息如下所述，全部错误信息请参　错误码　档。

错误码	HTTP 状态码	描述
CofferAlreadyExists	409 Conflict	指定的保险箱已存在。
CofferAlreadyOwnedByYou	409 Conflict	指定的保险箱已存在且由当前账户创建。

错误码	HTTP 状态码	描述
AssumeRoleFailed	409 Conflict	扮演 失败，检查是否已授予保险箱 。
KMSFailed	409 Conflict	调 KMS 失败。

示例

请求

```
PUT /examplecoffer-1250000000 HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Sun, 26 May 2019 14:51:38 GMT
Content-Length: 207
Authorization: [Auth String]
Connection: close

{
"Locations": {
"Location": ["ap-tianjin"]
},
"EncryptType": 0,
"KeyId": "aaa-bbb-ccc",
"KeyRegion": "ap-beijing"
}
```

响应

```
HTTP/1.1 200 OK
Content-Length: 0
Connection: close
Date: Sun, 26 May 2019 14:51:37 GMT
Server: tencent-cdcs
x-cdcs-request-id: NWNlYWE3ZjlfZDQyNzVkNjRfMzg1N18yNzFh****
```

删除保险箱

最近更新时间：2023-02-03 11:06:12

接口描述

- **接口名称**：DeleteCoffer
- **接口功能**：该请求接口用于删除指定的保险箱。该接口的请求者需要对保险箱有写入权限。

> **注意：**
> 删除保险箱前，请确保保险箱内的数据已全部过期，且未完成上传的分块数据已全部清空，否则会无法删除保险箱。

请求

请求示例

```
DELETE /<CofferName-APPID> HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 的详情，请参见 请求签名 文档。

请求参数

此接口无请求参数。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求体

此接口无请求体。

响应

响应头

此接口仅返回公共响应头部，详情请参见 公共响应头部 文档。

响应体

此接口响应体为空。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述
CofferNotEmpty	409 Conflict	保险箱不为空。
NoSuchCoffer	404 Not Found	指定的保险箱不存在。

示例

请求

```
DELETE /examplecoffer-1250000000 HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Tue, 28 May 2019 03:19:13 GMT
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 204 No Content
Content-Length: 0
Connection: close
Date: Tue, 28 May 2019 03:19:14 GMT
Server: tencent-cdcs
x-cdcs-request-id: NWNlY2E4YjFfNjljMDBiMDlfMmNiZTlfZGE0****
```

查询保险箱列表

最近更新时间：2023-02-03 11:06:16

接口描述

- **接口名称：** GetService
- **接口功能：** 该接　用于查询请求者名下的数据保险箱列表。

请求

请求示例

```
GET / HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String **的详情，请参**　请求签名　**档。**

请求参数

此接　**请求参数。**

请求头

此接　仅使　公共请求头部，详情请参　公共请求头部　**档。**

请求体

此接　**请求体。**

响应

响应头

此接　仅返回公共响应头部，详情请参　公共响应头部　**档。**

响应体

查询成功，返回 application/json 数据，包含所有或特定地域下的保险箱列表。

```
{
"Owner": {
"ID": "string",
"DisplayName": "string"
},
"Coffers": {
"Coffer": [{
"Name": "string",
"Locations": {
"Location": ["Enum1"]
},
"CreationDate": "date",
}, {
"Name": "string",
"Locations": {
"Location": ["Enum1"]
},
"CreationDate": "date",
}]
}
}
```

请求参数描述如下：

节点名称（关键字）	类型	描述
Owner	Object	保险箱持有者信息。
Coffers	Object	保险箱列表。

Owner 对象的描述：

节点名称（关键字）	类型	描述

节点名称（关键字）	类型	描述
ID	string	保险箱持有者的完整 ID，格式为 qcs::cam::uin/[OwnerUin]:uin/[OwnerUin]。 例如 qcs::cam::uin/100000000001:uin/100000000001
DisplayName	string	保险箱持有者的名字。

Coffer 对象的描述：

节点名称（关键字）	类型	描述
Name	string	保险箱的名称，格式为 <CofferName-APPID>。 例如 examplecoffer-1250000000
Locations	Object	保险箱所在地域列表。
CreationDate	date	保险箱的创建时间，为 ISO8601 格式。 例如2019-05-24T105640Z

Locations 对象的描述：

节点名称（关键字）	类型	描述
Location	Enum	保险箱所在地域，枚举值请参 地域和访问域名 档。 例如 ap-beijing，ap-hongkong，eufrankfurt 等

错误码

此接口无特殊错误信息，全部错误信息请参 错误码 档。

示例

请求

```
GET / HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Fri, 24 May 2019 11:59:50 GMT
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 805
Connection: close
Date: Fri, 24 May 2019 11:59:51 GMT
Server: tencent-cdcs
x-cdcs-request-id: NWNlN2RjYjdfOGFiMjM1MGFfNTVjMl8zMmI1****

{
"Owner": {
"ID": "1234",
"DisplayName": "1234"
},
"Coffers": {
"Coffer": [{
"Name": "xh1592016477536259000-251010571",
"Locations": {
"Location": ["ap-tianjin"]
},
"CreationDate": "2020-06-13T10:47:58+08:00"
}, {
"Name": "xh1592016665413463000-251010571",
"Locations": {
"Location": ["ap-tianjin"]
},
"CreationDate": "2020-06-13T10:51:05+08:00"
}, {
"Name": "xh1592017330996120000-251010571",
"Locations": {
"Location": ["ap-tianjin"]
},
"CreationDate": "2020-06-13T11:02:11+08:00"
}, {
"Name": "xh1592017433508017000-251010571",
"Locations": {
```

```
"Location": ["ap-tianjin"]
}
"CreationDate": "2020-06-13T11:03:54+08:00"
}]
}
}
```

文件管理
获取文件列表

最近更新时间：2023-02-03 11:05:44

接口描述

- **接口名称**：GetCoffer
- **接口功能**：该请求接口可以列出数据保险箱内的部分或者全部对象。该接口的请求者需要对数据保险箱有读取权限。

> **说明：**
> 如果您在保险箱中上传了一个对象，并立即调用 GetCoffer 接口，由于此接口的最终一致性特性，返回的结果中可能不会包含您刚上传的对象。

请求

请求示例

```
GET /<CofferName-APPID> HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 的详情，请参见 请求签名 文档。

请求参数

名称	是否必选	类型	描述
prefix	否	string	对象键匹配前缀，限定响应中只包含指定前缀的对象键。
encoding-type	否	string	规定返回值的编码方式，可选值：url，代表返回的对象键为 URL 编码（百分号编码）后的值，例如“腾讯云”将被编码为%E8%85%BE%E8%AE%AF%E4%BA%91。

名称	是否必选	类型	描述
marker	否	string	起始对象键标记，从该标记之后（不含）按照 UTF-8 字典序返回对象键条目。
max-keys	否	integer	单次返回最大的条目数量，默认值为1000，最大为1000。 **注意：**该参数会限制每一次 List 操作返回的最大条目数，数据保险箱在每次 List 操作中将返回不超过 max-keys 所设定数值的条目。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求体

此接口无请求体。

响应

响应头

此接口返回公共响应头部

响应体

查询成功，返回 application/json 数据，包含保险箱中的对象信息。不同场景下的响应体请参见下方的实际案例。

```
{
"Name": "string",
"EncodingType": "string",
"Prefix": "string",
"Marker": "string",
"MaxKeys":integer,
"IsTruncated": "boolean",
"NextMarker": "string",
"EncryptType": interger,
"KeyId": "string",
"KeyRegion": "string",
"Locations": {
"Location": ["Enum"]
},
```

```
"Contents": [{
"Key": "string",
"LastModified": "date",
"ETag": "string",
"Size": integer,
"Owner": {
"ID": "string",
"DisplayName": "string"
}
}]
}
```

响应参数描述如下：

节点名称（关键字）	类型	描述
Name	string	保险箱的名称，格式为<CofferName-APPID>，例如examplecoffer-1250000000。
EncryptType	int	加密类型: • 0 = 不加密 • 1 = 使用公共主密钥 • 2 = 使用 KMS 主密钥
KeyId	string	主密钥的 ID，仅当 EncryptType = 2时，返回该参数。
KeyRegion	string	主密钥的地域，仅当 EncryptType = 2时，返回该参数。
Locations	Object	保险箱所在地域列表。
Location	Enum	保险箱所在地域，枚举值请参见 地域和访问域名 文档。 例如ap-beijing，ap-hongkong，eu-frankfurt等
EncodingType	string	编码格式，对应请求中的 encoding-type 参数，且仅当请求中指定了 encoding-type 参数才会返回该节点。
Prefix	string	对象键匹配前缀，对应请求中的 prefix 参数。
Marker	string	起始对象键标记，从该标记之后（不含）按照 UTF-8 字典序返回对象键条目，对应请求中的 marker 参数。

节点名称（关键字）	类型	描述
MaxKeys	integer	单次响应返回结果的最大条目数量，对应请求中的 max-keys 参数。 **注意：**该参数会限制每一次 List 操作返回的最大条目数，CDCS 在每次 List 操作中将返回不超过 max-keys 所设定数值的条目。如果由于您设置了 max-keys 参数，导致单次响应中未列出所有对象，CDCS 会返回一项 nextmarker 参数作为您下次 List 请求的入参，以便您后续进行列出对象。
IsTruncated	boolean	响应条目是否被截断，布尔值，例如 true 或 false。
NextMarker	string	仅当响应条目有截断（IsTruncated 为 true）才会返回该节点，该节点的值为当前响应条目中的最后一个对象键，当需要继续请求后续条目时，将该节点的值作为下一次请求的 marker 参数传入。
Contents	Object	对象条目。

Contents 对象的描述：

节点名称（关键字）	类型	描述
Key	string	对象键。
LastModified	date	对象创建时间，为 ISO8601 格式，如 2019-05-24T10:56:40Z。
ETag	string	对象的实体标签（Entity Tag），是对象被创建时标识对象内容的信息标签，可用于检查对象的内容是否发生变化， 例如“8e0b617ca298a564c3331da28dcb50df”，此头部并不一定返回对象的 MD5 值，而是根据对象上传和加密方式而有所不同。
Size	integer	对象大小，单位为 Byte。
Owner	Container	对象持有者信息。

Owner 对象的描述：

节点名称（关键字）	类型	描述
ID	string	对象持有者的 AppID。
DisplayName	string	对象持有者的名称。

错误码

此接口无特殊错误信息，全部错误信息请参见 错误码 文档。

示例

示例1：简单案例

请求

```
GET /examplecoffer-1250000000 HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Mon, 27 May 2019 11:26:14 GMT
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1517
Connection: close
Date: Mon, 27 May 2019 11:26:15 GMT
Server: tencent-cdcs
x-cdcs-coffer-region: ap-beijing
x-cdcs-request-id: NWNlYmM5NTdfZjI4NWQ2NF81ZmMwX2Q5N2E1****

{
"Name": "examplecoffer-1250000000",
"EncodingType": "",
"Prefix": "",
"Marker": "",
"MaxKeys": 1000,
"IsTruncated": "false",
"NextMarker": "",
"EncryptType": 0,
"KeyId": "",
"KeyRegion": "",
"Locations": {
```

```
"Location": ["ap-tianjin"]
},
"Contents": [{
"Key": "test1.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}, {
"Key": "test2.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}, {
"Key": "test3.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 90,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}]
}
```


示例2：需分页时获取第一页

请求

```
GET /examplecoffer-1250000000 HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Mon, 27 May 2019 11:07:30 GMT
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 321247
Connection: close
Date: Mon, 27 May 2019 11:07:30 GMT
Server: tencent-cdcs
x-cdcs-coffer-region: ap-beijing
x-cdcs-request-id: NWNlYmM0ZjJfZDcyNzVkNjRfNjQ5OV9lNzdk****

{
"Name": " examplecoffer -1250000000",
"EncodingType": "",
"Prefix": "",
"Marker": "",
"MaxKeys": 1000,
"IsTruncated": "true",
"NextMarker": "",
"EncryptType": 0,
"KeyId": "",
"KeyRegion": "",
"Locations": {
"Location": ["ap-tianjin"]
},
"Contents": [{
"Key": "test1.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
```

```
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}, {
"Key": "test2.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}

...

, {
"Key": "test3.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}]
}
```

示例3：需分页时获取后续页

请求

```
GET /examplecoffer-1250000000?marker=example-object-1000.jpg HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: Mon, 27 May 2019 11:08:36 GMT
```

```
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1834
Connection: close
Date: Mon, 27 May 2019 11:08:36 GMT
Server: tencent-cdcs
x-cdcs-coffer-region: ap-beijing
x-cdcs-request-id: NWNlYmM1MzRfZmVhODBiMDlfMmViNjRfZDQw****

{
"Name": " examplecoffer -1250000000",
"EncodingType": "2",
"Prefix": "",
"Marker": " test1.html ",
"MaxKeys": 1000,
"IsTruncated": "true",
"NextMarker": " test7.html ",
"EncryptType": 0,
"KeyId": "aaaa-bbbb-cccc",
"KeyRegion": "ap- tianjin ",
"Locations": {
"Location": ["ap-tianjin"]
},
"Contents": [{
"Key": "test1.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
```

```
}, {
"Key": "test2.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}

...

, {
"Key": "test3.html",
"LastModified": "2020-06-13T15:40:15+08:00",
"ETag": "\"f797031f3210ce6494466d619610926c\"",
"Size": 20,
"Owner": {
"ID": "1250000000",
"DisplayName": "1250000000"
}
}]
}
```

下载文件

最近更新时间：2020-06-09 10:19:34

接口描述

- **接口名称**：GetObject
- **接口功能**：该接口可以将数据保险箱中的对象（Object）下载至本地，该接口的请求者需要对目标对象有读取权限。

请求

请求示例

```
GET /<CofferName-APPID>/<ObjectKey> HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求参数

此接口无请求参数。

请求头

此接口除使用公共请求头部外，还支持以下请求头部，了解公共请求头部详情，请参见 公共请求头部 文档。

名称	是否必选	类型	描述
Range	否	string	RFC 2616 中定义的字节范围，范围值必须使用 bytes = first - last 格式，first 和 last 都是基于0开始的偏移量。 例如 bytes = 0 - 9，表示下载对象的开头10个字节的数据，此时返回 HTTP 状态码206（Partial Content）及 Content-Range 响应头部，如果不指定，则表示下载整个对象。

请求体

此接口无请求体。

响应

响应头

此接口除返回公共响应头部外，还返回以下响应头部，了解公共响应头部详情请参见 公共响应头部 文档。

名称	类型	描述
Content-Range	string	RFC 2616 中定义的返回内容的字节范围，仅当请求中指定了 Range 请求头部时，才会返回该头部。

响应体

此接口请求的响应体为对象（文件）内容。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述
AssumeRoleFailed	409 Conflict	扮演角色失败，检查是否已授予保险箱角色。
KMSFailed	409 Conflict	调用 KMS 失败。

获取文件信息

最近更新时间：2020-06-09 10:19:39

接口描述

- **接口名称**：HeadObject
- **接口功能**：该接口用于判断指定对象是否存在及有权限，该接口的请求者需要对目标对象有读取权限。

请求

请求示例

```
HEAD /<CofferName-APPID>/<ObjectKey> HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 的详情，请参见 请求签名 文档。

请求参数

此接口无请求参数。

请求头

此接口无特殊请求头部信息，了解公共请求头部详情，请参见 公共请求头部 文档。

请求体

此接口无请求体。

响应

响应头

此接口无特殊响应头部信息，还返回以下响应头部，了解公共响应头部详情，请参见 公共响应头部 文档。

响应体

此接口响应体为空。

错误码

此接口无特殊错误信息，全部错误信息请参见 错误码 文档。

上传文件

最近更新时间：2020-06-09 10:19:44

接口描述

- **接口名称**：PutObject
- **接口功能**：该接口可以将本地的对象（Object）上传至指定数据保险箱中，该接口的请求者需要对数据保险箱有写入权限。

> 说明：
>
> - PutObject 接口最大支持上传5GB文件。如需上传大于5GB的文件，请使用分块上传 InitiateMultipartUpload 接口。
> - 如果请求头的 Content-Length 值小于实际请求体（body）中传输的数据长度，数据保险箱仍将成功创建文件，但对象大小只等于 Content-Length 中定义的大小，其他数据将被丢弃。
> - 如果试图添加已存在的同名对象，则接口会返回对象已存在的错误。

请求

请求示例

```
PUT /<CofferName-APPID>/<ObjectKey> HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Datex
Content-Type: Content Type
Content-Length: Content Length
Content-MD5: MD5
Authorization: Auth String

[Object Content]
```

> 说明：
>
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求参数

此接口无请求参数。

请求头

此接口无特殊请求头部，公共请求头部详情，请参见 公共请求头部 文档。

请求体

此接口请求的请求体为对象（文件）内容。

响应

响应头

此接口无特殊响应头，公共响应头部详情请参见 公共响应头部 文档。

响应体

此接口响应体为空。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述
ObjectAlreadyExists	409 Conflict	指定的对象已存在（也可能是正在上传中或分片上传中）。
ObjectAlreadyExistsOwnedByYou	409 Conflict	指定的对象已存在且由当前账户创建（也可能是正在上传中或分片上传中）。
AssumeRoleFailed	409 Conflict	扮演角色失败，检查是否已授予保险箱角色。
KMSFailed	409 Conflict	调用 KMS 失败。

生命周期管理
设置生命周期

最近更新时间：2023-02-03 11:05:11

接口描述

- **接口名称**：PutCofferLifecycle
- **接口功能**：该接口用于为数据保险箱创建一个新的生命周期配置，以设置过期时间。如果该数据保险箱已配置生命周期，使用该接口创建新配置的同时则会覆盖原有配置。

请求

请求示例

```
PUT /<CofferName-APPID>?lifecycle HTTP/1.1
Host: service.cdcs.myqcloud.com
Content-Length: length
Date: GMT Date
Authorization: Auth String
Content-MD5: MD5
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作的实现使用公共请求头部，详情请参见 公共请求头部 文档。

非公共头部

> **注意：**
> 以下头部为必选头部。

该请求操作的实现使用如下必选头部：

名称	是否必选	类型	描述
Content-MD5	是	String	RFC 1864中定义的经过 Base64 编码的请求体内容 MD5 哈希值，用于完整性检查，验证请求体在传输过程中是否发生变化。

请求体

该 API 接口请求的请求体具体节点内容为：

```
{
"Rule": [{
"Expiration": {
"Days": 30
}
}]
}
```

具体内容描述如下：

节点名称（关键字）	是否必选	类型	描述
Rule	是	Object	规则描述。
Expiration	否	Object	规则过期属性。
Days	否	Integer	指明对象在上传后多少天删除：该字段有效值为正整数，最大支持3650天。

响应

响应头

此接口仅返回公共响应头部，详情请参见 公共响应头部 文档。

响应体

该响应体返回为空。

错误码

以下描述此请求可能会发生的特殊且常见的错误情况，具体的错误原因可参考返回的 message 进行排查。获取更多关于数据保险箱的错误码信息，或者产品所有的错误列表，请参见 错误码 文档。

错误码	HTTP 状态码	描述
NoSuchCoffer	404 Not Found	访问的数据保险箱不存在。
MalformedXML	400 Bad Request	XML 格式不合法。
InvalidArgument	400 Bad Request	请求参数不合法。

示例

请求

```
PUT /examplecoffer-1250000000?lifecycle HTTP/1.1
Host:service.cdcs.myqcloud.com
Date: Wed, 16 Aug 2017 11:59:33 GMT
Authorization: [Auth String]
Content-MD5: LcNUuow8OSZMrEDnvndw1Q==
Content-Length: 348
Content-Type: application/x-www-form-urlencoded

{
"Rule": [{
"Expiration": {
"Days": 10
}
}]
}
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Date: Wed, 16 Aug 2017 11:59:33 GMT
```

```
Server: tencent-cdcs
x-cdcs-request-id: NTk5NDMzYTRfMjQ4OGY3Xzc3NGRfMWY=
```

获取生命周期

最近更新时间：2023-02-03 11:05:15

接口描述

- **接口名称**：GetCofferLifecycle
- **接口功能**：该接口用于查询数据保险箱的生命周期配置，如果该数据保险箱没有配置生命周期规则，会返回NoSuchLifecycleConfiguration。

请求

请求示例

```
GET /<CofferName-APPID>?lifecycle HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization：Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作的实现使用公共请求头部，了解公共请求头详情，请参见 公共请求头部 文档。

非公共头部

该请求操作无特殊的请求头部信息。

请求体

该请求的请求体为空。

响应

响应头

公共响应头

该响应包含公共响应头部，了解公共响应头部详情，请参见 公共响应头部 文档。

特有响应头

该响应无特殊的响应头。

响应体

响应体中各个元素的内容及含义与 PutCofferLifecycle 时的请求体一致。详情请参见 创建生命周期 文档中的请求体节点描述内容。

错误码

该请求操作无特殊错误信息，常见的错误信息请参见 错误码 文档。

示例

请求

```
GET /examplecoffer-1250000000?lifecycle HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Wed, 16 Aug 2017 12:23:54 GMT
Authorization: [Auth String]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 312
Date: Wed, 16 Aug 2017 12:23:54 GMT
Server: tencent-cdcs
x-cdcs-request-id: NTk5NDM5NWFfMjQ4OGY3Xzc3NGRfMjA=


{
"Rule": [{
"Expiration": {
"Days": 10
}
```

```
}]
}
```

访问策略
查看访问策略

最近更新时间：2023-02-03 11:04:40

接口描述

- **接口名称**：GetCofferPolicy
- **接口功能**：该接口可以向数据保险箱读取权限策略。

请求

请求示例

```
GET /<CofferName-APPID>?policy HTTP/1.1
Host:service.cdcs.myqcloud.com
Date:date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作的实现使用公共请求头部，了解公共请求头部详情请参见 公共请求头部 文档。

非公共头部

该请求操作无特殊的请求头部信息。

请求体

该请求的请求体为空。

响应

响应头

公共响应头

该响应包含公共响应头部，了解公共响应头详情请参见 公共响应头部 文档。

特有响应头

该响应无特殊的响应头。

响应体

响应返回权限策略。

```
{
"Statement": [
{
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
},
"Effect": "allow",
"Action": [
"name/cdcs:GetCoffer"
],
"Resource": [
"qcs::cdcs:ap-guangzhou:uid/1250000000:examplecoffer-1250000000/*"
]
}
],
"version": "2.0"
}
```

错误码

该请求操作无特殊错误信息，常见的错误信息请参见 错误码 文档。

示例

请求

```
GET /examplecoffer-1250000000?policy HTTP/1.1
Host:service.cdcs.myqcloud.com
Authorization: [Auth String]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 237
Connection: keep-alive
Date: Thu Jan 19 16:21:46 2017
Server: tencent-cdcs
x-cdcs-request-id: NTg4MDc3MWFfOWIxZjRlXzZmNDVfZTBl

{
"Statement": [
{
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
},
"Effect": "allow",
"Action": [
"name/cdcs:GetCoffer"
],
"Resource": [
"qcs::cdcs::uid/1250000000:examplecoffer-1250000000/*"
]
}
],
"version": "2.0"
}
```

设置访问策略

最近更新时间：2023-02-03 11:04:47

接口描述

- **接口名称**：PutCofferPolicy
- **接口功能**：该接口可以向数据保险箱写入权限策略，当数据保险箱已存在权限策略时，该请求上传的策略将覆盖原有的权限策略。

请求

请求示例

```
PUT /<CofferName-APPID>?policy HTTP/1.1
Host: service.cdcs.myqcloud.com
Date: date
Content-Type:application/json
Content-MD5:MD5
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作使用公共请求头部实现，了解公共请求头部详情，请参见 公共请求头部 文档。

请求参数

无特殊请求参数。

请求体

下面的示例表示给主账号100000000001下的子账号100000000011授权，以允许子账号访问保险箱名为examplecoffer-1250000000 中的对象列表。关于访问策略中的元素介绍及授权策略示例，请参见 访问策略语言概述。

```
{
"Statement": [
{
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000011"
]
},
"Effect": "allow",
"Action": [
"name/cdcs:GetCoffer"
],
"Resource": [
"qcs::cdcs:ap-guangzhou:uid/1250000000:examplecoffer-1250000000/*"
]
}
],
"version": "2.0"
}
```

响应

响应头

公共响应头

该响应使用公共响应头，了解公共响应头详情，请参见 公共响应头部 文档。

特有响应头

该请求操作无特殊的响应头部信息。

响应体

该请求响应体为空。

错误码

无返回特殊错误码，一般常见错误码，请参见 错误码 文档。

示例

请求

```
PUT /examplecoffer-1250000000?policy HTTP/1.1
Host: service.cdcs.myqcloud.com
Authorization: [Auth String]
Content-Type: application/json
Content-Length: 233

{
"Statement": [
{
"Principal": {
"qcs": [
"qcs::cam::uin/100000000001:uin/100000000001"
]
},
"Effect": "allow",
"Action": [
"name/cdcs:GetCoffer"
],
"Resource": [
"qcs::cdcs:ap-guangzhou:uid/1250000000:examplecoffer-1250000000/*"
]
}
],
"version": "2.0"
}
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu Jan 19 16:19:22 2017
Server: tencent-cdcs
x-cdcs-request-id: NTg4MDc2OGFfNDUyMDRlXzc3NTlfZTc4
```

清除访问策略

最近更新时间：2023-02-03 11:04:51

接口描述

- **接口名称**：DeleteCofferPolicy
- **接口功能**：该接口可以向数据保险箱删除权限策略，只有数据保险箱所有者有权限发起该请求。如果权限策略不存在，将返回200 OK。

请求

请求示例

```
DELETE /<CofferName-APPID>?policy HTTP/1.1
Host:service.cdcs.myqcloud.com
Date: date
Authorization: Auth String
```

> 说明：
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作的实现使用公共请求头，了解公共请求头部详情，请参见 公共请求头部 文档。

非公共头部

该请求操作无特殊的请求头部信息。

请求体

该请求的请求体为空。

响应

响应头

公共响应头

该响应包含公共响应头部，了解公共响应头部详情，请参见 公共响应头部 文档。

特有响应头

该响应无特殊的响应头部。

响应体

该响应体为空。

示例

请求

```
DELETE /examplecoffer-1250000000?policy HTTP/1.1
Host:service.cdcs.myqcloud.com
Authorization: [Auth String]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Thu Jan 19 16:30:21 2017
Server: tencent-cdcs
x-cdcs-request-id: NTg4MDc5MWRfNDQyMDRlXzNiMDRfZTEw
```

分片上传
舍弃一个分块上传并删除已上传的块

最近更新时间：2023-02-03 11:03:48

接口描述

- **接口名称**：AbortMultipartUpload
- **接口功能**：该接口用于实现舍弃一个分块上传并删除已上传的块。当您调用 AbortMultipartUpload 时，如果有正在使用 UploadParts 上传块的请求，则 UploadParts 会返回失败。当该 UploadId 不存在时，会返回 404 NoSuchUpload。

> **注意：**
> 建议您及时完成分块上传或者舍弃分块上传，因为已上传但是未终止的块会占用存储空间，进而产生存储费用。

请求

请求示例

```
DELETE /<CofferNane-APPID>/<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求参数

具体内容如下：

参数名称	必选	类型	描述

参数名称	必选	类型	描述
uploadId	是	String	标识本次分块上传的 ID。 使用 InitiateMultipartUpload 接口初始化分片上传时会得到一个 uploadId，该 ID 不但唯一标识这一分块数据，也标识了这分块数据在整个文件内的相对位置。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求体

该请求的请求体为空。

响应

响应头

此接口仅返回公共响应头部，详情请参见 公共响应头部 文档。

响应体

该请求的响应体为空。

示例

请求

```
DELETE /examplecoffer-1250000000/exampleobject?uploadId=1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b08e HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Tue, 26 Oct 2013 21:22:00 GMT
Authorization: [Auth String]
```

响应

```
HTTP/1.1 204 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
```

```
Date: Tue, 26 Oct 2013 21:22:00 GMT
Server: tencent-cdcs
x-cdcs-request-id: NTg3ZjI5MzlfOTgxZjRlXzZhYjNfMjBh
```

完成整个分块上传

最近更新时间：2023-02-03 11:03:53

接口描述

- **接口名称**：CompleteMultipartUpload
- **接口功能**：接口请求用来实现完成整个分块上传。当使用 UploadParts 上传所有分块完成后，必须调用该 API 来完成整个文件的分块上传。

> ⚠ 注意：
> 在使用该 API 时，您必须在请求 Body 中给出每一个块的 PartNumber 和 ETag，用来校验块的准确性。

由于分块上传完成后需要合并，而合并需要数分钟时间，因而当合并分块开始时，数据保险箱会立即返回200的状态码，在合并的过程中，数据保险箱会周期性的返回空格信息来保持连接活跃，直到合并完成，数据保险箱会在 Body 中返回合并完成后的整个块内容。

- 当上传的分块小于1MB的时候，在调用该 API 时，会返回400 EntityTooSmall。
- 当上传块编号不连续的时候，在调用该 API 时，会返回400 InvalidPart。
- 当请求 Body 中的块信息没有按序号从小到大排列，在调用该 API 时，会返回400 InvalidPartOrder。
- 当 UploadId 不存在的时候，在调用该 API 时，会返回404 NoSuchUpload。

> ⚠ 注意：
> 建议您及时完成分块上传或者舍弃分块上传，因为已上传但是未终止的块会占用存储空间，进而产生存储费用。

请求

请求示例

```
POST /<CofferName-APPID>/<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Content-length: Size
Authorization: Auth String
```

> 说明：
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求体

该 API 接口请求的请求体具体节点内容为：

```
{
"Part": [{
"PartNumber": 1,
"ETag": "827ccb0eea8a706c4c34a16891f84e7b"
}, {
"PartNumber": 2,
"ETag": "927ccb0eea8a706c4c34a16891f84e8b"
}]
}
```

请求参数描述如下：

节点名称（关键字）	是否必选	类型	描述
Part	是	Object	用来说明本次分块上传中每个块的信息。

Part 对象的描述：

节点名称（关键字）	是否必选	类型	描述
PartNumber	是	Integer	块编号。
ETag	是	String	每个块文件的 MD5 算法校验值。

响应

响应头

该响应仅使用公共响应头部，了解公共响应头详情，请参见 公共响应头部 文档。

响应体

该响应体返回为 **application/json** 数据，包含完整节点数据的内容展示如下：

```
{
"Locations": {
"Location": ["ap-beijing"]
},
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
"ETag": "\"3a0f1fd698c235af9cf098cb74aa25bc\""
}
```

请求参数描述如下：

节点名称（关键字）	类型	描述
Locations	Object	新创建的对象的所在地域列表。
Location	Enum	新创建的对象的所在地域，枚举值请参见 地域和访问域名 文档。例如 ap-beijing，ap-hongkong，eu-frankfurt 等
Coffer	String	分块上传的目标保险箱，格式为 CofferName-APPID，例如：examplecoffer-1250000000。
Key	String	对象名称。
ETag	String	合并后对象的唯一标签值，该值不是对象内容的 MD5 校验值，仅能用于检查对象唯一性。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述
AssumeRoleFailed	409 Conflict	扮演角色失败，检查是否已授予保险箱角色。
KMSFailed	409 Conflict	调用 KMS 失败。

示例

请求

```
POST /examplecoffer-1250000000/exampleobject?uploadId=1484728886e63106e87d8207536ae8521c89c42a436fe23bb58854a7bb5e87b7d77d4ddc48 HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Wed，18 Jan 2017 16:17:03 GMT
Authorization: [Auth String]
Content-Length: 138

{
"Part": [{
"PartNumber": 1,
"ETag": "827ccb0eea8a706c4c34a16891f84e7b"
}, {
"PartNumber": 2,
"ETag": "927ccb0
eea8a706c4c34a16891f84e8b"
}]
}
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 277
Connection: keep-alive
Date: Wed，18 Jan 2017 16:17:03 GMT
Server: tencent-cdcs
x-cdcs-request-id: NTg3ZjJlMjVfNDYyMDRlXzM0YzRfMjc1

{
"Locations": {
"Location": ["ap-beijing"]
},
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
```

```
"ETag": "\"3a0f1fd698c235af9cf098cb74aa25bc\""
}
```

初始化分片上传

最近更新时间：2023-10-08 09:17:34

接口描述

- **接口名称**：InitiateMultipartUpload
- **接口功能**：该接口请求用于实现初始化分片上传，成功执行此请求后将返回 UploadId，用于后续的 UploadPart 请求。如果已存在同名的对象，则接口会返回对象已存在的错误。创建成功后，除非取消上传，否则将无法上传或分片上传同名的对象。

请求

请求示例

```
POST /<CofferName-APPID>/<ObjectKey>?uploads HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作的实现使用公共请求头部，了解公共请求头详情，请参见 公共请求头部 文档。

非公共头部

推荐头部

名称	必选	类型	描述
Content-Type	否	String	RFC 2616 中定义的内容类型（MIME），将作为 Object 元数据保存。

请求体

该请求的操作请求体为空。

响应

响应头

公共响应头

该响应使用公共响应头，了解公共响应头详情，请参见 公共响应头部 文档。

特有响应头

响应体

该响应体返回为 application/json 数据，包含完整节点数据的内容展示如下：

```
{
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
"UploadId": "1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b08e"
}
```

具体的数据内容如下：

节点名称（关键字）	类型	描述
Coffer	string	分片上传的目标 Coffer，由用户自定义字符串和系统生成 AppID 数字串组成，例如 examplecoffer-1250000000。
Key	string	Object 的名称。
UploadId	string	在后续上传中使用的 ID。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述

错误码	HTTP 状态码	描述
ObjectAlreadyExists	409 Conflict	指定的对象已存在（也可能是正在上传中或分片上传中）。
ObjectAlreadyExistsOwnedByYou	409 Conflict	指定的对象已存在且由当前账户创建（也可能是正在上传中或分片上传中）。
AssumeRoleFailed	409 Conflict	扮演角色失败，检查是否已授予保险箱角色。
KMSFailed	409 Conflict	调用 KMS 失败。

示例

请求

```
POST /examplecoffer-1250000000/exampleobject?uploads HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Fri, 10 Mar 2016 09:45:46 GMT
Authorization: [Auth String]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 230
Connection: keep-alive
Date: Fri, 10 Mar 2016 09:45:46 GMT
Server: tencent-cdcs
x-cdcs-request-id: NTg3ZjIzZTZfOWIxZjRlXzZmMzhfMWRj

{
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
"UploadId": "1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb
```

```
9633b08e"
}
```

查询正在进行中的分块上传任务

最近更新时间：2023-02-03 11:04:03

接口描述

- **接口名称**：ListMultipartUploads
- **接口功能**：该接口用于查询正在进行中的分块上传任务，单次请求操作最多列出1000个正在进行中的分块上传，该请求需要有数据保险箱的读权限。

请求

请求示例

```
GET /<CofferName-APPID>?uploads HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求参数

具体内容如下：

名称	必选	类型	描述
encoding-type	否	String	规定返回值的编码格式，合法值：url。
prefix	否	String	限定返回的 Object key 必须以 Prefix 作为前缀。 注意使用 prefix 查询时，返回的 key 中仍会包含 Prefix。
max-uploads	否	String	设置最大返回的 multipart 数量，合法取值从1到1000，默认1000。

名称	必选	类型	描述
upload-id-marker	否	String	起始对象键标记，从该标记之后（不含）按照 UTF-8 字典序返回对象键条目。

请求体

该请求的请求体为空。

响应

响应头

此接口仅返回公共响应头部，详情请参见 公共响应头部 文档。

响应体

该响应体返回为 application/json 数据，包含完整节点数据的内容展示如下：

```
{
"Coffer": "",
"EncodingType": "",
"UploadIdMarker": "",
"IsTruncated": "",
"NextUploadIdMarker": "",
"Prefix": "",
"MaxUploads": 0,
"Upload": [{
"Key": "",
"UploadId": "",
"Initiator": {
"ID": "",
"DisplayName": ""
},
"Owner": {
"ID": "",
"DisplayName": ""
},
"Initiated": ""
```

```
}]
}
```

请求参数描述如下：

节点名称（关键字）	类型	描述
Coffer	String	分块上传的目标 Coffer，由用户自定义字符串和系统生成 APPID 数字串由中划线连接而成，例如examplecoffer-1250000000。
Encoding-Type	String	规定返回值的编码格式，合法值：url。
UploadIdMarker	String	列出条目从该 Marker 开始。
NextUploadIdMarker	String	假如返回条目被截断，则返回的 NextMarker 就是下一个条目的起点。
MaxUploads	String	设置最大返回的 multipart 数量，合法取值从0 - 1000。
IsTruncated	Boolean	返回条目是否被截断，布尔值：TRUE，FALSE。
Prefix	String	限定返回的 Objectkey 必须以 Prefix 作为前缀，注意使用 prefix 查询时，返回的 key 中仍会包含 Prefix。
Upload	Object	每个 Upload 的信息。

Upload 对象的描述：

节点名称（关键字）	类型	描述
Key	String	Object 的名称。
UploadID	String	标示本次分块上传的 ID。
Initiator	Object	用来表示本次上传发起者的信息。
Owner	Object	用来表示这些分块所有者的信息。
Initiated	Date	分块上传的起始时间。

Initiator 对象的描述：

节点名称（关键字）	类型	描述

节点名称（关键字）	类型	描述
ID	String	用户唯一的 CAM 身份 ID。
DisplayName	String	用户身份 ID 的简称（UIN）。

Owner 对象的描述：

节点名称（关键字）	描述	类型
ID	String	用户唯一的 CAM 身份 ID。
DisplayName	String	用户身份 ID 的简称（UIN）。

错误码

以下描述此请求可能会发生的一些特殊的且常见的错误情况：

错误码	HTTP 状态码	描述
InvalidArgument	400 Bad Request	max-uploads 必须是整数，且值介于0 - 1000之间，否则返回 InvalidArgument。 encoding-type 只能取值 url，否则会返回 InvalidArgument。

获取更多关于数据保险箱的错误码信息，或者产品所有的错误列表，请参见 错误码 文档。

示例

请求

```
GET /examplecoffer-1250000000?uploads HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Wed, 18 Jan 2015 21:32:00 GMT
Authorization: [Auth String]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
```

```
Content-Length: 1203
Date: Wed, 18 Jan 2015 21:32:00 GMT
Server: tencent-cdcs
x-cdcs-request-id: NTg3ZjI0ZGRfNDQyMDRlXzNhZmRfMjRl

{
"Coffer": "examplecoffer-1250000000",
"EncodingType": "",
"UploadIdMarker": "",
"IsTruncated": "false",
"NextUploadIdMarker": "",
"Prefix": "",
"MaxUploads": 1000,
"Upload": [{
"Key": "Object",
"UploadId": "1484726657932bcb5b17f7a98a8cad9fc36a340ff204c79bd2f51e7dddf0b6d1da62
20520c",
"Initiator": {
"ID": "qcs::cam::uin/100000000001:uin/100000000001",
"DisplayName": "100000000001"
},
"Owner": {
"ID": "qcs::cam::uin/100000000001:uin/100000000001",
"DisplayName": "100000000001"
},
"Initiated": "Wed Jan 18 16:12:38 2017"
}]
}
```

查询特定分块上传中的已上传的块

最近更新时间：2023-02-03 11:04:09

接口描述

- **接口名称**：ListParts
- **接口功能**：该接口用于查询特定上传分块中已上传的分块，即罗列出指定 UploadId 所属的所有已上传成功的分块。

请求

请求示例

```
GET /<CofferName-APPID>/<ObjectKey>?uploadId=UploadId HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Authorization: Auth String
```

> **说明：**
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

此接口仅使用公共请求头部，详情请参见 公共请求头部 文档。

请求参数

名称	必选	类型	描述
UploadId	是	string	标识本次分块上传的 ID。使用 InitiateMultipartUpload 接口 初始化分片上传时会得到一个 uploadId，该 ID 不但唯一标识这一分块数据，也标识了这分块数据在整个文件内的相对位置。
encoding-type	否	string	规定返回值的编码方式。
max-parts	否	string	单次返回最大的条目数量，默认1000。

名称	必选	类型	描述
part-number-marker	否	string	默认以 UTF－8 二进制顺序列出条目，所有列出条目从 part-number-marker 开始。

请求体

该请求的请求体为空。

响应

响应头

此接口仅返回公共响应头部，详情请参见 公共响应头部 文档。

响应体

查询成功，返回 application/json 数据，包含已完成的分片信息。


```
{
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
"UploadId": "14846420620b1f381e5d7b057692e131dd8d72dfa28f2633cfbbe4d0a9e8bd07199
33545b0",
"Initiator": {
"ID": "1250000000",
"DisplayName": "1250000000"
},
"Owner": {
"ID": "qcs::cam::uin/100000000001:uin/100000000001",
"DisplayName": "100000000001"
},
"PartNumberMarker": "1",
"IsTruncated": "true",
"NextPartNumberMarker": "1",
"MaxParts": 0,
"Part": [{
"PartNumber": 1,
"LastModified": "Tue Jan 17 16:43:37 2017",
```

```
"ETag": "\"a1f8e5e4d63ac6970a0062a6277e191fe09a1382\"",
"Size": 5242880
}]
}
```

请求参数描述如下：

节点名称（关键字）	类型	描述
Coffer	string	分块上传的目标数据保险箱，数据保险箱的名字由用户自定义字符串和系统生成 AppID 数字串连接而成，例如：examplecoffer-1250000000。
Encoding-Type	string	编码格式。
Key	string	Object 的名字。
UploadId	string	标识本次分块上传的 ID。
Initiator	Object	用来表示这些分块所有者的信息。
Owner	Object	用来表示这些分块所有者的信息。
PartNumberMarker	string	默认以 UTF - 8 二进制顺序列出条目，所有列出条目从 marker 开始。
NextPartNumberMarker	string	假如返回条目被截断，则返回 NextMarker 就是下一个条目的起点。
MaxParts	string	单次返回最大的条目数量。
IsTruncated	boolean	响应请求条目是否被截断，布尔值：true 或 false。
Part	Object	元数据信息。

Initiator 对象的描述：

节点名称（关键字）	类型	描述
ID	string	创建者的一个唯一标识。
DisplayName	string	创建者的用户名描述。

Owner 对象的描述：

节点名称（关键字）	类型	描述
ID	string	创建者的唯一标识。
DisplayName	string	创建者的用户名描述。

Part 对象的描述：

节点名称（关键字）	类型	描述
PartNumber	string	块的编号。
LastModified	string	说明块最后被修改时间。
ETag	string	块的 MD-5 算法校验值。
Size	string	说明块大小，单位是 Byte。

示例

请求

```
GET /examplecoffer-1250000000/exampleobject?uploadId=1585130821cbb7df1d11846c073ad648e8f33b087cec2381df437acdc833cf654b9ecc6361 HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Wed, 25 Mar 2020 10:07:25 GMT
Authorization: [Auth String]
Connection: close
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 1119
Connection: close
Date: Wed, 25 Mar 2020 10:07:25 GMT
Server: tencent-cdcs
x-cdcs-request-id: NWU3YjJkNWRfMjNhZjJhMDlfNWY5Ml8zMmUy****
```

```
{
"Coffer": "examplecoffer-1250000000",
"Key": "exampleobject",
"UploadId": "14846420620b1f381e5d7b057692e131dd8d72dfa28f2633cfbbe4d0a9e8bd07199
33545b0",
"Initiator": {
"ID": "1250000000",
"DisplayName": "1250000000"
},
"Owner": {
"ID": "qcs::cam::uin/100000000001:uin/100000000001",
"DisplayName": "100000000001"
},
"PartNumberMarker": "1",
"IsTruncated": "true",
"NextPartNumberMarker": "1",
"MaxParts": 0,
"Part": [{
"PartNumber": 1,
"LastModified": "Tue Jan 17 16:43:37 2017",
"ETag": "\"a1f8e5e4d63ac6970a0062a6277e191fe09a1382\"",
"Size": 5242880
}]
}
```

将对象按照分块的方式上传到保险箱

最近更新时间：2023-02-03 11:04:14

接口描述

- **接口名称**：UploadPart
- **接口功能**：该接口请求用于实现将对象按照分块的方式上传到数据保险箱，最多支持10000分块，每个分块大小为1MB - 5GB，最后一个分块可以小于1MB。

> 说明：
>
> i. 分块上传首先需要进行初始化，使用 InitiateMultipartUpload 接口 实现，初始化后会得到一个 uploadId ，唯一标识本次上传。
> ii. 在每次请求 UploadPart 时，需要携带 partNumber 和 uploadId，partNumber 为块的编号，支持乱序上传。
> iii. 当传入 uploadId 和 partNumber 都相同的时候，后传入的块将覆盖之前传入的块。当 uploadId 不存在时会返回404错误，NoSuchUpload。

请求

请求示例

```
PUT /<CofferName-APPID>/<ObjectKey>?partNumber=PartNumber&uploadId=UploadId HTTP/1.1
Host: cdcs.<Region>.myqcloud.com
Date: GMT Date
Content-Length: Size
Authorization: Auth String

[Object]
```

> 说明：
>
> Authorization: Auth String 详情，请参见 请求签名 文档。

请求头

公共头部

该请求操作使用公共请求头部实现，了解公共请求头部详请，请参见 公共请求头部 文档。

非公共头部

必选头部

该请求操作需要请求头使用必选头部，具体内容如下：

名称	是否必选	类型	描述
Content-Length	是	String	RFC 2616 中定义的 HTTP 请求内容长度（字节）。

推荐头部

该请求操作推荐请求头使用推荐头部，具体内容如下：

名称	是否必选	类型	描述
Content-MD5	否	String	RFC 1864 中定义的经过 Base64 编码的请求体内容 MD5 哈希值，用于完整性检查，验证请求体在传输过程中是否发生变化。

请求参数

具体内容如下：

参数名称	是否必选	类型	描述
partNumber	是	String	标识本次分块上传的编号，partNumber 需要大于等于1。
uploadId	是	String	标识本次分块上传的 ID，使用 InitiateMultipartUpload 接口初始化分片上传时，会得到一个 uploadId，该 ID 不但唯一标识这一分块数据，也标识了这分块数据在整个文件内的相对位置。

请求体

该请求的请求体为该分块的数据内容。

响应

响应头

公共响应头

该响应包含公共响应头，了解公共响应头详情，请参见 公共响应头部 文档。

特有响应头

该响应无特有响应头部信息。

响应体

该请求的响应体为空。

错误码

此接口的特殊错误信息如下所述，全部错误信息请参见 错误码 文档。

错误码	HTTP 状态码	描述
AssumeRoleFailed	409 Conflict	扮演角色失败，检查是否已授予保险箱角色。
KMSFailed	409 Conflict	调用 KMS 失败。

示例

请求

```
PUT /examplecoffer-1250000000/exampleobject?partNumber=1&uploadId=1484727270323ddb949d528c629235314a9ead80f0ba5d993a3d76b460e6a9cceb9633b08e HTTP/1.1
Host: cdcs.ap-beijing.myqcloud.com
Date: Wed，18 Jan 2017 16:17:03 GMT
Authorization: [Auth String]
Content-Length: 10485760

[Object]
```

响应

```
HTTP/1.1 200 OK
Content-Type: application/xml
Content-Length: 0
Connection: keep-alive
Date: Wed，18 Jan 2017 16:17:03 GMT
Etag: "e1e5b4965bc7d30880ed6d226f78a5390f1c09fc"
Server: tencent-cdcs
x-cdcs-request-id: NTg3ZjI0NzlfOWIxZjRlXzZmNGJfMWYy
```

错误码

最近更新时间：2021-11-24 10:48:01

概述

本文将为您介绍请求出错时，返回的错误码和对应错误信息。

错误响应

Content-Type：application/xml

对应 HTTP 状态码：3XX、4XX、5XX。

响应体

```
{
"Code": "string",
"Message": "string",
"Resource": "string",
"RequestId": "string",
"TraceId": "string"
}
```

具体参数描述如下：

Error 对象的描述：

节点名称（关键字）	类型	描述
Code	string	错误码，用来定位唯一的错误条件和确定错误场景，具体请参见 错误码列表
Message	string	具体的错误信息。
RequestId	string	每次请求发送时，服务端将会自动为请求生成一个 ID，遇到问题时，该 ID 能更快地协助数据保险箱定位问题。
TraceId	string	每次请求出错时，服务端将会自动为这个错误生成一个 ID，遇到问题时，该 ID 能更快地协助数据保险箱定位问题。
Resource	string	资源名称。

错误码列表

错误码	HTTP 状态码	描述
InternalError	500 Internal Server Error	服务器内部错误。
MethodNotAllowed	405 Method Not Allowed	请求 Method 错误。
CofferNotFound	404 Not Found	保险箱不存在。
ObjectNotExists	404 Not Found	文件不存在。
MultipartNotExists	404 Not Found	分片不存在。
NoSuchUpload	404 Not Found	该分片上传不存在
EntityTooSmall	400 Bad Request	上传分片过小
InvalidPart	400 Bad Request	上传分片编号不连续
InvalidPartOrder	400 Bad Request	上传分片顺序错误
InvalidArgument	400 Bad Request	参数错误。
InvalidCofferName	400 Bad Request	保险箱名字不符合规范。
InvalidObjectKey	400 Bad Request	文件名不符合规范。
InvalidObjectPrefix	400 Bad Request	文件前缀格式错误。
InvalidLifeCycle	400 Bad Request	生命周期格式错误。
CofferAlreadyExists	409 Conflict	保险箱已经存在。

错误码	HTTP 状态码	描述
CofferAlreadyOwnedByYou	409 Conflict	保险箱已经创建。
CofferNotEmpty	409 Conflict	保险箱不为空，无法删除。
NoSuchCoffer	404 Not Found	指定的保险箱不存在。
MultipartAlreadyCompleted	409 Conflict	分片上传已完成。
ObjectAlreadyUploading	409 Conflict	已经有对象在上传。
ObjectAlreadyExistsOwnedByYou	409 Conflict	指定的对象已存在且由当前账户创建（也可能是正在上传中或分片上传中）。
ObjectAlreadyExists	409 Conflict	对象已经存在。
AssumeRoleFailed	409 Conflict	获取授权失败。
KMSFailed	409 Conflict	调用 KMS 发生错误。
AuthFailed	401 Unauthorized	鉴权失败。
ObjectEmpty	400 Bad Request	文件为空。
DataSynchronization	409 Conflict	数据正在同步中，可以稍后重试。
AlreadyLock	409 Conflict	记录已经被锁定，可以稍后重试。
PolicyCheckFailed	401 Unauthorized	策略鉴权失败。
PolicyFormatCheckFailed	409 Conflict	策略格式错误。
CofferCountExceedsLimit	409 Conflict	用户名下的保险箱数量超标。
ServiceNotOpen	401 Unauthorized	用户不在白名单中。
ConsoleAuthFailed	401 Unauthorized	控制台鉴权失败。