**TCS-SDK使用手册**

1. 简介 2
   1. TCS 2
   2. TCS-SDK 2
2. Web接入 2
   1. 接入方法 2
   2. 参数约定 2
   3. UI定制 3
   4. 范例 4
3. API接入 4
   1. 返回码 4
   2. 图片上传 5
   3. 音频上传 6
   4. 视频上传 6
   5. 其他文件上传 6
4. 业务处理规则 6
   1. 图片处理规则 6
   2. 音频处理规则 8
   3. 视频处理规则 8
5. 返回约定 8
   1. 图片返回约定 8
   2. 音频返回约定 9
   3. 视频返回约定 9
6. 参考附录 9
7. 简介
   1. TCS

TCS全称该亚云存储（TalCloud Storage）服务，借助其他云存储平台（目前主要是百度BOS），解决公司所有产品线静态文件存储和访问加速的问题。

除此之外，GCS还提供其他业务特性：图片和音/视频的业务处理。其中，图片业务包括图片上传接入、多尺寸的图片文件生成，以及图片尺寸和加载速度优化等。音/视频业务则包括上传接入、基本信息获取（播放时长、截屏）、转码，以及音/视频压缩等等。

* 1. TCS-SDK

TCS-SDK主要提供文件（图片、音/视频等）的上传接入，接入方式目前有2种：一种是Web方式接入；另一种是API方式接入。

1. Web接入

为了降低接入成本，让需求方尽可能少做工作，Web方式接入向需求方提供JSSDK。其实质是通过引用外部js动态创建iframe对象，并在iframe中引入外部页面，实现文件上传的统一接入。同时，为了满足需求方对UI的个性定制要求，也向需求方开放了部分UI的定制方法。

* 1. 接入方法

Web接入方式有两个文件需要引入（测试环境）：

CSS：在static/multi-media/jssdk下

引入CSS文件，可以为需求方提供默认的UI；如果需要定制部分UI，则不需要引入。

JS：static/multi-media/jssdk下

引入JS文件后，即可通过js代码创建上传对象TCS.Uploader，如：

new TCS.Uploader(function(data){

alert(JSON.stringify(data));

}, { clientId: 'ABCDEFGHIJK' });

* 1. 参数约定

TCS.Uploader对象的构造方法有三个参数，原型如下：

TCS.Uploader([containerId,] callback, options);

参数说明：

|  |  |
| --- | --- |
| **参数名** | **描述** |
| containerId | 容器元素的ID，可选。  传入该参数时，Uploader对象会把iframe嵌入在由容器元素ID指定的元素内。需求方可以通过对容器元素进行UI定制，实现部分UI定制的需要。 |
| callback | 用于接收上传结果的回调方法，必选。  callback原型：function callback(data)。  参数data是Uploader返回的JSON对象，格式如下：  {  code: 返回码,  message: "返回描述",  files: [  { name: "原文件名", url: "上传之后的url" }  ]  } |
| options | options是一个JSON对象，用于指定更多的参数选项，必选。  options对象的格式如下：  {  clientId: '', // 客户端ID  type: 'image', // 上传的业务类型  title: '上传文件', // 默认UI的标题  center: true // 默认UI是否居中显示  } |

options属性说明：

|  |  |
| --- | --- |
| **属性** | **描述** |
| clientId | 客户端ID，必选。  每个需要接入Web上传的应用（甚至业务），都必须申请GCS的ClientId，并在上传的时候，传入该ID以作验证。 |
| type | 上传业务的类型，可选；默认为"image"。  目前支持4种类型：  image 图片  audio 音频  video 视频  misc 其他类型  *要说明的是，misc类型上传的文件仅提供存储服务，不会做任何业务处理；而其他三种类型，均有业务处理，例如图片类型的，上传之后会根据业务需要生成各种规格尺寸的图片。* |
| title | 默认UI（即未指定容器ID的情况）的标题，可选；  默认是"上传文件"。 |
| center | 默认UI是否居中显示，可选；默认为true。 |

* 1. UI定制

目前开放的UI定制，仅针对容器元素的UI，即iframe以及iframe的容器可定制UI。有两种方式：

直接在构造方法指定容器元素的ID；这种方法的定制相对更全面一些，在定制UI的同时，还能接管部分用户交互流程。比如：上传界面的显示、隐藏，以及可以定义遮罩等等。

使用默认UI，但重新定义CSS。即以默认的css文件为参考，重新定义css属性，从而实现UI的定制；但这种方法无法改变默认交互。

* 1. 范例

<head>

<!-- 引入默认UI的css文件 -->

<link href=“http://tcs-en-beta.speiyou.com/multi-media/jssdk/tcs.upload.css" type="text/css" rel="stylesheet" />

</head>

…

<input id="upi" type="button" value="图片" onclick="javascript:upi();" />

<input id="upii" type="button" value="定制" onclick="javascript:upii();" />

<div id="up" style="width: 500px; height: 200px;"></div>

<!-- 引入JSSDK -->

<script src=“http://tcs-en-beta.speiyou.com/multi-media/jssdk/tcs.upload.js” type="text/javascript"></script>

<script type="text/javascript">

// 创建上传对象，使用默认UI

var ui = new TCS.Uploader(function(data){

alert(JSON.stringify(data));

}, { clientId: 'ABCDEFGHIJK' });

function upi() { ui.upload(); }

// 创建上传对象，指定容器元素ID

var uii = new TCS.Uploader('up', function(data){

alert(JSON.stringify(data));

}, { clientId: 'ABCDEFGHIJK' });

function upii() { uii.upload(); }

</script>

更详细的范例，请访问:<http://tcs-en-beta.speiyou.com/multi-media/upload.html>

1. API接入
   1. 返回码

|  |  |
| --- | --- |
| **返回码** | **描述** |
| 0 | 服务器处理成功。 |
| 90001 | 不支持的上传业务类型。  目前仅支持4种：image、audio、video和misc。一般情况，要么是传错了参数，要么是调错了API。 |
| 90002 | 上传FORM的enctype不是multipart/form-data。 |
| 90003 | 上传失败。  一般情况是上传数据不完整，或服务器IO错误等等。 |
| 90004 | 数据不足。  一般都是上传FORM中找不到要求的数据导致的。clientId、domain和FILE都是必要数据；同时，FILE必须是文件，且可以是多个。 |
| 90005 | 验证失败。  clientId和domain不正确或不匹配导致的。调用API时，必须严格按GCS给出的clientId和domain进行调用。 |

* 1. 图片上传

API名：upload/image

URL：<http://tcs-en.speiyou.com/>multi-media/upload/image

请求方式：POST，enctype="multipart/form-data"

参数：

|  |  |
| --- | --- |
| **参数名** | **描述** |
| clientId | 客户端ID，每个需要接入API上传的应用（甚至业务），都必须申请GCS的ClientId，并在上传的时候，传入该ID以作验证（可以和Web接入使用同一个ClientId）。 |
| domain | API接入授权的域名。 |
| FILE | 需要上传的图片文件，允许有多个文件上传。 |
| 以下参数用于优化图片处理业务，即客户端可以指定上传的图片是属于哪个业务的；详细规则见5.1。 | |
| FILE\_o | 上传原图优化图。 |
| FILE\_rWsxHszWlxHl | 上传原比例缩放图。 |
| FILE\_rWxH | 上传指定尺寸图片。 |
| FILE\_{url} | 上传原图。 |

*\* clientId和domain用于验证接入授权信息（类似其他云平台的AppKey和SecretKey）。*

返回：

|  |  |
| --- | --- |
| **返回值** | **描述** |
| code | 0：成功 |
| message | / |
| files | File数组，详见File。 |

File解释：

|  |  |
| --- | --- |
| **属性** | **描述** |
| name | 原文件名 |
| url | 文件上传之后的url |

* 1. 音频上传

API名：upload/audio

URL：<http://tcs-en.speiyou.com/>multi-media/upload/audio

参数和返回格式，均同图片上传API。

* 1. 视频上传

API名：upload/video

URL：<http://tcs-en.speiyou.com/>multi-media/upload/video

参数和返回格式，均同图片上传API。

* 1. 其他文件上传

API名：upload/misc

URL：<http://tcs-en.speiyou.com/>multi-media/upload/msic

参数和返回格式，均同图片上传API。

1. 业务处理规则

GCS允许需求方在后台定义不同类型的文件上传之后的业务处理。例如可以指定一张图片上传之后，需要同时生成一系列的不同规格尺寸的图片，以适用于不同场景的图片展示。但在需求方的自定义业务规则之上，还有一些系统内置的默认处理规则；这些规则通常是更偏向功能实现的底层规则，用以更好的对不规则图片进行兼容处理。这些规则通常也是不允许经常变更的。

* 1. 图片处理规则

图片的默认处理规则有三种：原图优化、原比例缩放，以及指定尺寸输出。

* 原图优化

把原图进行文件大小的优化；原图如果有1M，优化后在100~300k。

* 原比例缩放

通过指定目标图片的最大和最小宽高，将原图按原比例进行缩放。

这种方式并不直接给出目标尺寸，而是设定一个输出尺寸的范围，无论宽高，在尺寸范围内即可。因此，目标图片的宽高其实存在一个比例值范围，如果原图比例在目标范围之间，则认为是合适的比例；否则，视为不合适。与此同时，原图尺寸与目标尺寸之间，还存在另外三种关系：比目标图片最小尺寸小（S）、在目标图片尺寸范围内（M），以及比目标图片最大尺寸大（L）。

|  |  |  |
| --- | --- | --- |
|  | **比例合适（I）** | **比例不适（O）** |
| 小（S） | IS | OS |
| 合适（M） | IM | OM |
| 大（L） | IL | OL |

根据表格，我们要处理6种规则：

|  |  |
| --- | --- |
| **差异编码** | **处理规则** |
| OS | 以短边为准，按比例缩放至同边方向的最小值，然后裁剪掉长边方向的多余部分。 |
| OL |
| OM |
| IS | 以短边为准，按比例缩放至同边方向的最小值。 |
| IM | 不做处理，原图输出。 |
| IL | 以长边为准，按比例缩放至同边方向的最大值。 |

* 指定尺寸输出

将原图处理成指定宽高的目标图片。

在介绍具体规则之前，我们先看看下面的表格：

|  |  |  |
| --- | --- | --- |
|  | **宽（W）** | **高（H）** |
| 小（S） | WS | HS |
| 合适（M）  目标尺寸的80%~200% | WM | HM |
| 大（L） | WL | HL |

我们对宽和高，以及原图和目标图片在宽高上的尺寸差异设定了阀值（80%~200%），并进行了编码。从而，我们可以通过一个简单的编码来标识原图和目标图片之间的差异。

例如：原图200x200，目标图片尺寸250x120，那么二者的差异编码是：WMHS。即：原图的宽与目标图片的宽，差异为合适（(200\*80%)~(200\*200%)=160~400）；原图的高与目标图片的高，差异是小。

根据上述方法，我们能得到9个差异编码，以下是9个差异编码的处理规则：

|  |  |
| --- | --- |
| **差异编码** | **处理规则** |
| WSHS | 补偿 |
| WSHM | 先裁剪（高），再补偿；  或直接补偿（高度不够） |
| WSHL | 先裁剪（高），再补偿 |
| WMHS | 先裁剪（宽），再补偿；  或直接补偿（宽度不够） |
| WMHM | 先缩放（不等比，按短边为依据进行缩放），再裁剪；  或直接缩放（等比） |
| WMHL | 先缩放（宽），再裁剪 |
| WLHS | 先裁剪（宽），再补偿 |
| WLHM | 先缩放（高），再裁剪 |
| WLHL | 先裁剪（根据指定尺寸按比例放大进行裁剪），再缩放 |

*\* 补偿，是指当尺寸不够时，用白色填充空余部分。*

* 业务信息输出

查询图片上传后的业务处理信息。

* 1. 音频处理规则

待定。

* 1. 视频处理规则

待定。

1. 返回约定
   1. 图片返回约定

无论是Web方式接入，还是API方式接入，图片上传后的返回url，始终是原图的url。因此，如果要访问特定业务处理之后的图片，我们需要进行另外的约定。根据4.1，我们知道图片的3种处理规则；基于此，我们进行以下约定：

* 原图优化

在原图的url后面增加“\_o”后缀。

例如，原图上传之后的url为：<http://bos.bj.com/aaa.png>，那么，原图优化的url为：<http://bos.bj.com/aaa.png_o>。

* 原比例缩放

在原图url后增加“\_rWsxHszWlxHl”后缀。

Ws：业务约束的最小宽度。

Hs：业务约束的最小高度。

Wl：业务约束的最大宽度。

Hl：业务约束的最大高度。

例如，原图上传之后的url为：<http://bos.bj.com/aaa.png>，业务约束的最小宽高是120，最大宽高分别是600和500，那么原比例缩放的图片url为：<http://bos.bj.com/aaa.png_r120x120z600x500>。

* 指定尺寸输出

在原图url后增加“\_rWxH”后缀。

W：指定尺寸的宽度。

H：指定尺寸的高度。

例如，原图上传之后的url为：<http://bos.bj.com/aaa.png>，指定输出的尺寸为800\*600，那么指定尺寸的图片url为：<http://bos.bj.com/aaa.png_r800x600>。

* 业务信息输出

在原图url后增加“\_meta”后缀。

图片的业务信息主要是指根据业务定义的一系列图片尺寸。

该文件是一个JSON文件，其格式如下：

{

"\_self" : { width: 0, height: 0 },

"\_o" : { width: 0, height: 0 },

"\_rWsxHszWlxHl" : { width: 0, height: 0 },

"\_rWxH" : { width: 0, height: 0 }

}

*\* 用js解析时，是一个object对象；在java中解析，是一个Map。*

|  |  |
| --- | --- |
| **属性** | **描述** |
| \_self | 原图尺寸。 |
| \_o | 原图优化后的尺寸。 |
| \_rWsxHszWlxHl | 原比例缩放的图片尺寸。 |
| \_rWxH | 指定输出尺寸的图片尺寸。 |

*\* 其中，\_rWsxHxzWlxHl和\_rWxH属性值是根据业务需要生成的，可以存在多条。*

* 1. 音频返回约定

待定。

* 1. 视频返回约定

待定。

1. 参考附录