Spark API 开发指南

版本: 2.3.0
日期: 2015-03-31

北京梦之窗数码科技有限公司

目 录

1. 概述

利用 Spark API 可以与 CC 视频 Spark 云进行对接，使用 Spark 云的主要视频功能。当前 Spark API 的版本具有上传、播放、获取视频信息、获取用户信息、删除视频、获取视频分类等接口。

目前 Spark API 仅对合作方开启，如果您有使用的需求，请通过 CC 视频后台申请 API Key，通过审核后才能使用 Spark API。

2. 通信约定

Spark API 的远程通信接口基于 HTTP 协议，并有以下约定：

2.1 HTTP方法

所有接口采用 GET 请求。

2.2 返回格式

接口的返回格式包括 XML 格式和 JSON 格式，编码均为 UTF-8。对于不同的接口，正确的返回结果的格式会在每个接口中单独定义，错误的返回结果具有统一的形式，如下所示：

```
<?xml version="1.0" encoding="UTF-8"?>
<error>ERROR_CODE</error>
```

```
{
   “error”:”ERROR_CODE”
}
```

实际中，下表中的某一个具体的错误码会替换掉上面的 ERROR_CODE：

错误码	说明
INVALID_REQUEST	用户输入参数错误
SPACE_NOT_ENOUGH	用户剩余空间不足
SERVICE_EXPIRED	用户服务已经过期
PROCESS_FAIL	服务器处理失败
TOO_MANY_REQUEST	访问过于频繁
PERMISSION_DENY	用户服务无权限

2.3 编码格式

Spark API 只接受 UTF-8 格式编码的信息，返回的数据也都是 UTF-8 编码的。当需要通过 GET 请求传递参数时，QueryString 里面的 value 值都需要进行 URL Encode 之后，再进行传递。

2.4 加密

所有的 HTTP 通信都是加密的，加密的核心思想是将原始的 QueryString 转换为和请求时间相关的 HashedQueryString，我们称这个加密算法为 THQS 算法。关于 THQS 算法的细节请参见附录 1。

2.5 接口访问次数限制

当某一个接口的访问频率在一分钟之内超过100次的时候，该用户的 API 功能将被禁用，之后所有请求都将失效。如果上述阈值无法满足您的正常需求的时候，可以联系 CC 客服申请提升 API 的请求频率。

3. HTTP接口

3.1 获取用户信息

通过该接口可以获取指定用户的账户信息，地址为：

http://spark.bokecc.com/api/user

需要传递以下参数：

参数	说明
userid	用户 id，不可为空
format	返回格式，xml 或 json

返回数据user包含如下字段：

字段名	说明
account	用户账户
version	版本信息
expired	到期时间
space	用户空间信息
traffic	用户流量信息

space包含如下字段：

字段名	说明
total	用户空间总量，单位G
remain	用户空间剩余总量，单位G
used	用户空间使用总量，单位G

traffic包含如下字段：

字段名	说明
total	用户流量总量，单位G
remain	用户剩余流量大小，单位G
used	用户已使用流量大小，单位G

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<user>
   <account>test@test.com</account>
   <version><![CDATA[试用版]]></version>
   <expired>2011-06-06</expired>
   <space>
      <total>2</total>
      <remain>1.9</remain>
      <used>0.1</used>
   </space>
   <traffic>
      <total>5</total>
      <remain>4.8</remain>
      <used>0.2</used>
   </traffic>
</user>
```

JSON格式的返回信息如下：

```
{
   "user":{
      "account":"test@test.com",
      "version":"试用版",
      "expired":"2011-06-06",
      "space":{
         "total":2,
         "remain":1.9,
         "used":0.1
      },
     "traffic":{
         "total":5,
         "remain":4.8,
         "used":0.2
      }
   }
}
```

3.2 获取视频信息

通过该接口可以获取指定用户的有效视频的信息，地址为：

http://spark.bokecc.com/api/video

需要传递以下参数：

参数	说明
userid	用户 id，不可为空
videoid	视频 id，不可为空
format	返回格式，xml 或 json

返回数据video包括如下字段：

字段名	说明
id	视频ID
title	视频标题
desp	视频描述
tags	视频标签
duration	视频时长，单位秒
category	视频分类ID
image	视频截图地址
imageindex	视频截图序号
image-alternate	视频截图排列信息

image-alternate包括如下字段：

字段名	说明
index	视频截图排列序号
url	视频截图地址

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<video>
    <id>01234567</id>
    <title><![CDATA[视频标题]]></title>
    <desp><![CDATA[视频描述]]></desp>
    <tags><![CDATA[标签1 标签2 标签3]]></tags>
    <duration>12345</duration>
    <category>12345</category>
    <image>http://image.bokecc.com/abc.jpg</image>
    <imageindex>1</imageindex>
    <image-alternate>
        <index>0</index>
        <url>http://image.bokecc.com/abc0.jpg</url>
    </image-alternate>
    <image-alternate>
        <index>1</index>
        <url>http://image.bokecc.com/abc1.jpg</url>
    </image-alternate>
    ...
</video>
```

JSON格式的返回信息如下：

```
{
   "video":{
      "id":"01234567",
      "title":"视频标题",
      "desp":"视频描述",
      "tags":"标签1 标签2 标签3",
      "duration":12345,
      "category":"12345",
      "image":"http://image.bokecc.com/abc.jpg",
      "imageindex":1,
      "image- alternate":[{
         "index":0,
         "url":"http://image.bokecc.com/abc0.jpg"
      },
      ...
      ]
   }
}
```


3.3 批量获取视频信息

通过该接口可以获取指定用户的一批有效视频(不包括删除、正在处理的视频)的信息，地址为:

http://spark.bokecc.com/api/videos

需要传递以下参数:

参数	说明
userid	用户 id，不可为空
videoid_from	起始 videoid，若为空，则从上传的第一个视频开始
videoid_to	终止 videoid，若为空，则到最后一个上传的视频
num_per_page	返回信息时，每页包含的视频个数 注:允许范围为 1~100
page	当前页码
format	返回格式，xml 或 json

返回数据videos包含以下字段:

字段名	说明
total	返回视频数量
video	视频信息

video包含以下字段:

字段名	说明
id	视频ID
title	视频标题
desp	视频描述
tags	视频标签
duration	视频时长，单位秒
category	视频分类ID
image	视频截图地址
imageindex	视频截图序号
image-alternate	视频截图排列信息

image-alternate包含以下字段：

字段名	说明
index	视频截图排列序号
url	视频截图地址

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<videos>
   <total>100</total>
   <video>
      <id>01234567</id>
      <title><![CDATA[Video Title]]></title>
      <desp><![CDATA[Video Description]]></desp>
      <tags><![CDATA[tag1 tag2 tag3]]></tags>
      <duration>12345</duration>
      <category>12345</category>
      <image>http://image.bokecc.com/abc.jpg</image>
      <imageindex>1</imageindex>
      <image-alternate>
         <index>0</index>
         <url>http://image.bokecc.com/abc0.jpg</url>
      </image-alternate>
      <image-alternate>
         <index>1</index>
         <url>http://image.bokecc.com/abc1.jpg</url>
      </image-alternate>
      ...
   </video>
   ...
</videos>
```

JSON格式的返回信息如下：

```
{
   "videos":{
      "total":100,
      "video":[
         {
            "id":"01234567",
            "title":"Video Title",
            "desp":"Video Description",
            "tags":"tag1 tag2 tag3",
            "duration":12345,
            "category":"12345",
            "image":"http://image.bokecc.com/abc.jpg",
            "imageindex":1,
            "image-alternate":[
               {
                  "index":0,
                  "url":"http://image.bokecc.com/abc0.jpg"
               },
               …
            ]
         },
         …
      ]
   }
}
```


3.4 获取视频播放代码

通过该接口可以获取指定视频的视频播放 HTML 代码段，地址为：

http://spark.bokecc.com/api/video/playcode

需要传递以下参数：

参数	说明
videoid	视频 id，不可为空
userid	用户 id，不可为空
playerid	播放器 id，若为空则返回默认播放器
player_width	播放器宽度，单位 px
player_height	播放器高度，单位 px
auto_play	是否自动播放，true 或 false
format	返回格式，xml 或 json

返回数据video包含以下字段：

字段名	说明
playcode	嵌入网页的播放代码

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<video>
    <playcode>
        <![CDATA[<script src='http://p.bokecc.com/player?
vid=96479767C315E6A9&siteid=1936D297411C3A27&autoStart=false&width=600&height=490&playerid
=6ED7421AB96B522E&playertype=1' type='text/javascript'></script>]]>
    </playcode>
</video>
```

JSON格式的返回信息如下：

```
{
    "video":{
        "playcode":"<script src='http://p.bokecc.com/player?
vid=96479767C315E6A9&siteid=1936D297411C3A27&autoStart=false&width=600&height=490&playerid
=6ED7421AB96B522E&playertype=1 type='text/javascript'></script>"
    }
}
```

3.5 编辑视频信息

通过该接口可以编辑指定视频的信息，地址为：

http://spark.bokecc.com/api/video/update

需要传递以下参数：

参数	说明
videoid	视频 id，不可为空
userid	用户 id，不可为空
title	视频标题
tag	视频标签
description	视频描述
categoryid	视频子分类 id
playurl	视频播放页面地址，如果不编辑播放地址，请勿加入此参数
imageindex	视频封面截图序号，如果不编辑封面截图，请勿加入此参数 注:只可编辑正常可播放状态的视频截图
format	返回格式，xml 或 json

返回数据video包含如下字段：

字段名	说明
id	视频ID
title	视频标题
desp	视频描述
tags	视频标签
category	视频分类ID
playurl	视频播放页面地址
imageindex	视频截图序号

XML格式的返回信息如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<video>
    <id>01234567</id>
    <title><![CDATA[Video Title]]></title>
    <desp><![CDATA[Video Description]]></desp>
    <tags><![CDATA[tag1 tag2 tag3]]></tags>
    <category>12345</category>
    <playurl>http://xxxx/1.html</playurl>
    <imageindex>1</imageindex>
</video>
```

JSON格式的返回信息如下:

```
{
    "video":{
        "id":"01234567",
        "title":"Video Title",
        "desp":"Video Description",
        "tags":"tag1 tag2 tag3",
        "category":"12345",
        "playurl":"http://xxxx/1.html",
        "imageindex":1
    }
}
```

3.6 删除视频

通过该接口可以删除指定视频的信息，地址为:

http://spark.bokecc.com/api/video/delete

需要传递以下参数:

参数	说明
videoid	视频 id，不可为空
userid	用户 id，不可为空
format	返回格式，xml 或 json

XML格式的返回信息如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<result>OK</result>
```

JSON格式的返回信息如下：

```
{
   "result":"OK"
}
```

3.7 获取视频分类

通过该接口可以获取指定用户创建的全部视频分类，地址为：

http://spark.bokecc.com/api/video/category

需要传递以下参数：

参数	说明
userid	用户 id，不可为空
format	返回格式，xml 或 json

注:返回结果中的 level 包括 BASIC 和 PREMIUM，BASIC 中只包含一个默认父分类，无法创建自定义父分类， PREMIUM 版本中包含用户创建的父分类。

返回数据video包含如下字段：

字段名	说明
level	视频分类等级
category	视频分类信息

category包含如下字段：

字段名	说明
id	分类ID
name	分类名称
sub-category	子分类信息

sub-category包含如下字段：

字段名	说明
id	子分类ID
name	子分类名称

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<video>
   <level>BASIC</level>
   <category>
      <id>123</id>
      <name><![CDATA[category1]]></name>
      <sub-category>
         <id>456</id>
         <name><![CDATA[sub category1]]></name>
      <sub-category>
      ...
   </category>
   ...
</video>
```

JSON格式的返回信息如下：

```
{
   "video":{
      "level":"PREMIUM",
      "category":[{
         "id":123,
         "name":"category1",
         "sub-category":[{
            "id":456,
            "name":"sub category1"
         },
         ...
         ]
      },
      ...
      ]
   }
}
```

3.8 搜索视频

通过该接口可以搜索指定信息的视频，地址为：

http://spark.bokecc.com/api/videos/search

需要传递以下参数：

参数	说明
userid	用户 id，不可为空
q	查询条件，不可为空 格式：查询字段:查询内容 查询字段：目前只支持TITLE 查询内容：查询关键字 注：格式中的":"为英文半角 Example：q=TITLE:test
sort	查询结果排序，不可为空 格式：排序字段:排序方式 排序字段：CREATION_DATE或FILE_SIZE 排序方式：ASC或DESC 注：格式中的":"为英文半角 Example：sort=CREATION_DATE:DESC
categoryid	视频子分类 id，如果不查询指定分类下的视频，请勿加入此参数
num_per_page	返回信息时，每页包含的视频个数 注:阈值为 1~100
page	当前页码
format	返回格式，xml 或 json

返回数据videos包含如下字段：

字段名	说明
total	返回视频数量
video	视频信息

video包含如下字段：

字段名	说明
id	视频ID
title	视频标题
desp	视频描述
tags	视频标签
duration	视频时长，单位秒
category	视频分类ID
creation-date	视频创建时间
filesize	视频文件大小，单位字节
image	视频截图地址

XML格式的返回信息如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<videos>
   <total>100</total>
   <video>
      <id>01234567</id>
      <title><![CDATA[Video Title]]></title>
      <desp><![CDATA[Video Description]]></desp>
      <tags><![CDATA[tag1 tag2 tag3]]></tags>
      <duration>12345</duration>
      <category>12345</category>
      <creation-date>2012-10-29 15:16:50</creation-date>
      <filesize>4174340</filesize>
      <image>http://image.bokecc.com/abc.jpg</image>
      ...
   </video>
   ......
</videos>
```

JSON格式的返回信息如下：

```
{
   "videos":{
      "total":100,
      "video":[
         {
            "id":"01234567",
            "title":"Video Title",
            "desp":"Video Description",
            "tags":"tag1 tag2 tag3",
            "duration":12345,
            "category":"12345",
            "creation-date":"2012-10-29 15:16:50",
            "filesize":4174340,
            "image":"http://image.bokecc.com/abc.jpg"
         },
         …
      ]
   }
}
```

4. 视频文件上传接口

4.1 Flash视频上传

Spark API 中所有的 Flash 接口需要 Flash 插件的版本在 10.1 以上才有效，使用前请确保 Flash 插件版本符合要求。

在上传视频的过程中，不用与 Spark 平台进行 HTTP 通信，使用 JavaScript 和 Spark 平台提供的 Flash 进行交互即可完成。关于如何在网页中嵌入 Flash 以及如何和 Flash 进行交互，请参阅附录 2。上传接口用到的所有 JavaScript 函数定义见下表:

函数定义	参数说明
on_spark_selected_file(file_name, file_size)	file_name: 上传文件名 file_size: 上传的文件大小，单位字节
on_spark_upload_validated(status, videoid)	status: 验证结果 videoid: 视频 id
on_spark_upload_progress(progress)	progress: 上传进度。正确时，0~100 之间的整数，包括 0 和 100;错误时，返回-1.

网页嵌入 Spark 平台提供的上传 Flash 时，需要传递下列参数:

Flash参数(flashvars)	说明
progress_interval	回调进度函数的时间间隔，默认 1 秒
notify_url	视频处理完毕后的通知地址

上传文件的过程一共分为四步:

第一步，选择文件。

Spark 平台提供一个透明的 Flash 进行文件上传，地址如下:

http://p.bokecc.com/flash/api/uploader.swf

由于它是透明的，所以可以置于任何一个 HTML 元素的上方而不影响的页面视觉效果。上传文件时，必须通过点击到该 Flash 从而打开浏览对话框进行文件选择。 文件选择成功后，Flash 会调用页面中的 on_spark_selected_file 函数，页面可以选择合适的方式处理该事件。

第二步，验证。

当用户选择文件后，需将下面的参数按照 THQS 算法处理后传递给 Flash 的 start_upload 函数后，才能开始上传流程。首先要进行参数和权限的验证，通过 后才开始文件上传。

参数	说明
userid	用户 id，不可为空
title	视频标题，若为空，则采用去后缀文件名作为 title
description	视频简介，可以为空
tag	视频标签，可以为空
categoryid	视频子分类 id，可选参数

验证完成后，Flash 会调用 on_spark_upload_validated 函数传递验证结果以及视频 id。验证状态码的含义如下:

验证状态码	说明
OK	成功
NETWORK_ERROR	网络错误
其它	Spark API 错误码

第三步，文件上传。

文件开始上传后，Flash 会周期性调用 on_spark_upload_progress 函数来报告上传进度，间隔秒数由 progress_interval 参数指定。如果上传的进度为负数， 则说明发生网络错误，上传中断。返回 100 则表示上传成功。当返回 100 或者 -1 后，Flash 就不再调用该函数了。

第四步，回调。

当 Spark 平台处理完毕视频后(上传、转码、审核都完成后)，会通过 HTTP 的 GET 方式通知你的网站。该地址由参数 notify_url 指定。通知时会以 THQS 方式携带以下参数:

参数	说明
videoid	视频 id，16位 hex 字符串
status	视频状态。”OK”表示视频处理成功，”FAIL”表示视频处理失败。
duration	片长(单位:秒)
image	视频截图地址

视频处理失败有多种情况，例如视频文件异常、视频内容违规等等。

```
<?xml version="1.0" encoding="UTF-8"?>
<result>OK</result>
```

当 notify_url 指定的接口返回上述 XML 时，Spark 平台会认为网站已经成功接收到了回调信息，不再进行重试。返回其它任何内容，Spark 平台会进行回调重试，重试的间隔会随着重试次数的增大而增大。若重试 7 次后，依然没有成功，则不再进行通知。因此，Spark 平台最多通知 8 次。这 8 次的通知时间距第一次的通知时间的差分别为:

[0，15s， 1m， 4m， 16m， 1h4m， 4h16m， 17h4m]

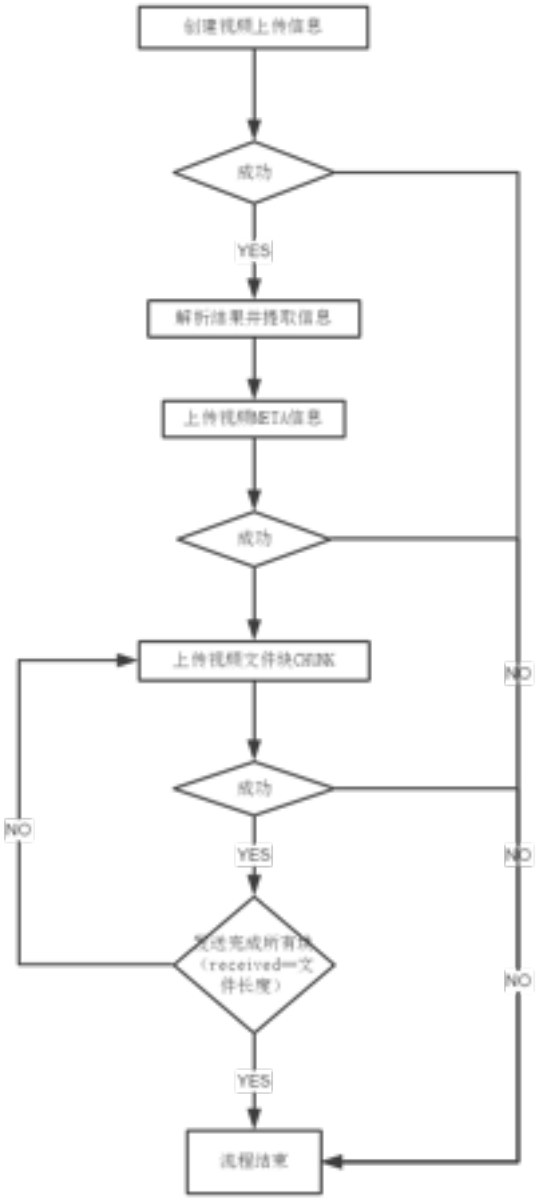
4.2 HTTP视频上传(断点续传)

此段文档用于指导使用CC视频云平台HTTP_API接口编码来完成上传视频文件功能。
userid 用户CC视频云平台账号
ccvid或vid 用户视频文件ID CC视频云平台指定
apikey 用户CC视频云平台API的密钥key

使用步骤

1 用户调用 创建视频上传信息 返回数据，并提取vid,metaurl,chunkurl等返回数据。
2 用户调用 上传视频META信息(metaurl) 将文件描述等信息发送到服务器。
3 用户反复调用 上传视频文件块CHUNK(chunkurl) 将文件实际数据发送到服务器直到全部发完。
4 如果调用 上传视频文件块CHUNK(chunkurl) 中间出现中断，可使用断点续传方式 上传视频META信息(metaurl) 获取已发文件长度后继续调用 上传视频文件块CHUNK(chunkurl) 将文件发送完成。

流程图



1　创建视频上传信息

通过该接口获取视频VID，上传URL等信息。
此接口为后续调用提供参数。需使用THQS算法将参数处理。

http://spark.bokecc.com/api/video/create
需要传递以下参数：

参数	含义
userid	用户id，不可为空
title	视频标题
tag	视频标签
description	视频描述
categoryid	二级分类id（主账号可选，子帐号不可为空）
filename	视频文件名
filesize	视频文件大小
notify_url	视频处理完毕的通知地址
format	返回格式：包含xml和json

返回信息，xml格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<uploadinfo>
    <videoid>3B35ED608474B8DB2BB</videoid>
    <userid>40EF004A6F1</userid>
    <servicetype>240593F08</servicetype>
    <metaurl><![CDATA[http://abc.com/servlet/uploadmeta]]></metaurl>
    <chunkurl><![CDATA[http://abc.com/servlet/uploadchunk]]> </chunkurl>
</uploadinfo>
```

返回信息，json格式如下：

```
{
  "uploadinfo": {
    "videoid": "3B35ED608474B8DB2BBA",
        "userid": “40EF004A6F",
        "servicetype": “240593F089",
        "metaurl": “http://abc.com/servlet/uploadmeta",
        "chunkurl": "http://abc.com/servlet/uploadchunk"
  }
}
```

异常返回格式

XML格式

```
<? xml version="1.0" encoding="UTF-8"?>
<error>SERVICE_EXPIRED</error>
```

JSON格式

```
{
    "error":" SERVICE_EXPIRED "
}
```

错误码的定义如下表：

错误码	含义
INVALID_REQUEST	用户输入参数错误
SPACE_NOT_ENOUGH	用户剩余空间不足
SERVICE_EXPIRED	用户服务终止
PROCESS_FAIL	服务器处理失败
TOO_MANY_REQUEST	访问过于频繁
PERMISSION_DENY	用户服务无权限

2　上传视频META信息

通过以下接口可以发送视频文件到平台接收服务器：

http://*.bokecc.com/servlet/uploadmeta

需要传递以下参数：

参数	含义
uid	用户id，不可为空　断点续传请求时为可选
ccvid	视频文件vid 不可为空
first	是否首次 first=1表示首次 断点续传为非首次 first=2
filename	视频名称　　断点续传请求时为可选
md5	视频文件md5　断点续传请求时为可选
filesize	视频文件大小 单位Byte　断点续传请求时为可选
servicetype	servicetype来源于 创建视频上传信息 接口 断点续传请求时为可选
format	返回格式：包含xml和json 可选 不选默认json返回

返回信息，内容统一

含义如下
result 结果　成功(0) 失败(-1)
msg　消息信息 错误描述
received　已接收文件长度　范围 0到文件大小

xml格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <result>0</result>
    <msg>success</msg>
    <received>0</received>
</root>
```

返回信息，json格式如下：

```
{
  "result": 0,
   "msg": “success",
   "received": 0
   }
```

异常返回格式

xml格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <result>-1</result>
    <msg>文件写入错误</msg>
    <received> – 1</received>
</root>
```

返回信息，json格式如下：

```
{
  "result": -1,
   "msg": “文件写入错误",
   "received": -1
   }
```

3　上传视频文件块CHUNK

http://*.bokecc.com/servlet/uploadchunk

需要传递以下参数：

参数	含义
ccvid	vid，视频id　不可为空
format	返回格式：包含xml和json　可选　不选默认json返回

返回信息，内容统一

含义如下
result 结果　成功(0) 失败(-1)

msg 消息信息 错误描述
received 已接收文件长度 范围 0到文件大小

xml格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <result>0</result>
    <msg>success</msg>
    <received>3413432</received>
</root>
```

返回信息，json格式如下：

```
    {
      "result": “0“,
       "msg": “success",
       "received": "412432"
       }
```

异常返回格式

xml格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <result>-1</result>
    <msg>文件写入错误</msg>
    <received>－1</received>
</root>
```

返回信息，json格式如下：

```
    {
      "result": “-1“,
       "msg": “文件写入错误",
       "received": “－1"
       }
```

此接口使用POST方式将实际文件的指定部分二进制数据发送到服务器，
服务器会返回已接收到的实际文件长度，当服务器返回文件长度等于发送文件长度时即文件已发送完成。

POST协议详解示例

HTTP协议需要用户修改 header中Content-Range内容。
Content-Range: bytes x-y/z
x 表示该段数据在文件中的开始位置==上次请求返回received数值
y 表示该段数据在文件中的结束位置
z 表示文件总长度
提醒：y表示文件索引下标位置，所以有 y <= z - 1
(binary)部分为文件部分块二进制内容

报文格式

```
POST /servlet/uploadchunk HTTP/1.1
Accept: text/*
Content-Type:multipart/form-data; boundary=----------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4)
Host: www.example.com
Content-Length: 421
Connection: Keep-Alive
Cache-Control: no-cache
Content-Range: bytes 0-100/500

------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7
Content-Disposition: form-data;name="file";filename="MyFile.jpg"
Content-Type: application/octet-stream

(binary)
------------Ij5GI3GI3ei4GI3ei4KM7GI3KM7KM7--
```

示例代码

Java版本代码
上传视频文件块CHUNK 接口3 示例代码

```
        /* url为/servlet/uploadchunk?ccvid=&format= */
        /* chunkStart为chunk起始位置 */
        /* chunkEnd为chunk结束位置 */
        /* file为文件 */
        /* bufferOut为实际文件输出二进制内容 */

        public String uploadchunk(String url, int chunkStart, int chunkEnd, File file, byte[] bufferOut) {
                HttpURLConnection conn = null;
                try {
                        String BOUNDARY = "---------CCHTTPAPIFormBoundaryEEXX" + new
Random().nextInt(65536); // 定义数据分隔线
                        URL openUrl = new URL(url);
                        conn = (HttpURLConnection)openUrl.openConnection();
                        // 发送POST请求必须设置如下两行
                        conn.setDoOutput(true);
                        conn.setDoInput(true);
                        conn.setUseCaches(false);
                        conn.setRequestMethod("POST");
                        conn.setRequestProperty("connection", "Keep-Alive");
                        conn.setRequestProperty("user-agent", "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_9_4)");
                        conn.setRequestProperty("Charsert", "UTF-8");
                        conn.setRequestProperty("Content-Type", "multipart/form-data; boundary=" +
BOUNDARY);
                        // content-range
                        conn.setRequestProperty("Content-Range", "bytes " + chunkStart + "-" + chunkEnd
+ "/" + file.length());

                        OutputStream out = new DataOutputStream(conn.getOutputStream());
                        StringBuilder sb = new StringBuilder();
                        sb.append("--").append(BOUNDARY).append("\r\n");
                        sb.append("Content-Disposition: form-data;name=\"file" + file.getName() +
"\";filename=\"" + file.getName()
                                        + "\"\r\n");
```

```
                sb.append("Content-Type: application/octet-stream\r\n");
                sb.append("\r\n");
                byte[] data = sb.toString().getBytes();
                out.write(data);
                out.write(bufferOut);
                out.write("\r\n".getBytes());                    // 定义最后数据分隔线
                byte[] end_data = ("--" + BOUNDARY + "--\r\n").getBytes();
                out.write(end_data);
                out.flush();
                out.close();

                // 定义BufferedReader输入流来读取URL的响应
                BufferedReader reader = new BufferedReader(new
InputStreamReader(conn.getInputStream()));
                StringBuffer resultBuf = new StringBuffer("");
                String line = null;
                while ((line = reader.readLine()) != null) {
                        resultBuf.append(line);
                }
                reader.close();
                conn.disconnect();
                return resultBuf.toString();
        } catch (Exception e) {
                System.out.println("发送POST请求出现异常！" + e);
                e.printStackTrace();
        } finally {
                if (conn != null)
                        conn.disconnect();
        }
        return null;
    }
```

附录 1. HTTP通信加密算法

当需要和 Spark 平台进行 HTTP 通信时，需要将原始的 Query String 转换为和请求时刻相关的 Hashed Query String 后再通过 GET 方法请求 Spark API。为了描述的方便，我们将 Query String 转换为 Hashed Query String 的算法称为 THQS 算法。在描述详细的算法流程之前，我们先介绍一下 Unix 时间戳的概念。

Unix 时间戳，即该时间到 1970 年 1 月 1 日(UTC/GMT 的午夜)之间的秒数。 例如，北京时间 2010 年 12 月 9 日 15 点 23 分 12 秒的 Unix 时间戳为 1291879392。

THQS 算法

假设原来的 QueryString 为 q， 通过以下 4 个步骤，即可得到最终用于通信的 HashedQueryString:

1. 对于q中的每个键值对按照键的字母顺序升序排序，得到排序后的请求字符串qs;
2. 加入当前时间的 Unix 时间戳和 Spark 平台帐号对应的 API Key 值，得到散列前的字符串 qf:
   qf ← qs&time=12345&salt=aSdF1234
3. 计算得到 qf 的 md5 值，假设为 abcdefg
   hash ← md5(qf)
4. 最终的 HashedQueryString 为:
   hqs ← qs&time=12345&hash=abcdefg

用 hqs 代替 q 进行 Http 通信。

下面举一个例子说明计算过程。假设用户从 Spark 获取到的 API Key 值是 aSdF1234， 当前时间为2010年12月9日15点23分12秒，原始的QueryString 是

name=harry&level=top&salary=1000

第一步，将上述 QueryString 按照字母顺序进行升序排序，结果是

level=top&name=harry&salary=1000

第二步，附加 time 值和 salt 值，得到取 hash 前的字符串

level=top&name=harry&salary=1000&time=1291879392&salt=aSdF1234

第三步，对上述字符串取 md5 值

hash=BF04A55B30CFF562F7ADD9F054AB7FFB

因此，最终进行 Http 通信的字符串为

level=top&name=harry&salary=1000&time=1291879392&hash=BF04A55B30CFF562F7ADD9F054AB7FFB

附录 2. Flash和JavaScript交互

Spark 平台的通信模式中，有若干种方式需要通过与页面中嵌入的 Flash 交互完成。 Flash 和页面有多种方式可以交互，为了保证 Spark 平台提供的 Flash 能够在各种环境下都能正常工作，推荐采用以下方式进行处理。

首先，需要用将 swf 文件嵌入到网页中，推荐采用 swfobject 1.5 版本。

将 swfobject.js 添加到网页中后，用以下 js 语句将 swf 嵌入到网页中，

```
<script type="text/javascript">
var swfobj=new SWFObject('http://xxx/xx.swf'， 'swfname'， '80'， '80'， '8');
swfobj.addVariable('title'，'test');
swfobj.addVariable('number'， 123);
swfobj.addParam('allowFullscreen'，'true');
swfobj.addParam('allowScriptAccess'，'always');
swfobj.addParam('wmode'，'transparent');
swfobj.write('divid');
</script>
```

其次，调用 Flash 中的函数的时候，如果 Flash 提供的函数名叫 func，而需要传递的参数是 param 的话，那么下面这句就可以调用该函数。

```
swfname["func"](param);
```

其中，swfname 是嵌入 Flash 的 id 值。