POLYV 视频上传 SDK

Polyv JavaScript 上传 SDK 为您提供上传媒体文件到保利威云点播平台的开发工具包。

功能

- 快捷上传多种格式的媒体文件。
- 支持上传时的各种设置，如文件标题、描述、标签、上传目录、是否开启课件优化处理等。
- 默认采用分片并发上传的方式，支持**断点续传**。

使用方法

前提条件

1. 使用本 SDK 前，要先开通**保利威云点播服务**。如果您还不了解该服务，请登录产品主页查看，详见：云点播。
2. 获取 secretKey 等相关信息用于用户身份校验，您可以在「云点播管理后台 -> 设置 -> API接口」页面中找到相关信息，点击这里登录后台。

浏览器支持

- IE(>=10)和Edge。
- 主流版本的 Chrome、Firefox、Safari。
- 以主流版本 Chrome 为核心的浏览器，如最新版本的 QQ 浏览器、360 浏览器等。

集成 SDK

您可以选择以下任意一种方法调用本 SDK：

方法一：引入在线资源

```
<!-- 指定版本 -->
<script src="//player.polyv.net/resp/vod-upload-js-sdk/1.2.3.3/vod-upload-js-sdk.min.js"></script>
<!-- 注意，1.4.0版本或后续的版本， SDK的静态资源地址有变化 -->
<!-- 1.4.0或后续的指定版本 -->
<script src="https://websdk.videocc.net/vod-upload-js-sdk/1.12.0/vod-upload-js-sdk.min.js"></script>
<!-- 最新版本(推荐使用) -->
<script src="https://websdk.videocc.net/vod-upload-js-sdk/latest/vod-upload-js-sdk.min.js"></script>
```

方法二：通过 npm 安装

第一步，在项目目录下运行安装命令：

```
npm install @polyv/vod-upload-js-sdk
```

第二步， 在页面中引入（需要构建工具支持）：

```
import PlvVideoUpload from '@polyv/vod-upload-js-sdk'

// 此npm包默认采用commonjs2方式，如在项目中引入报错的话可更改成
import PlvVideoUpload from "@polyv/vod-upload-js-sdk/vod-upload-js-sdk.min.js";
```

或者

```
const PlvVideoUpload = require('@polyv/vod-upload-js-sdk');

// 此npm包默认采用commonjs2方式，如在项目中引入报错的话可更改成
const PlvVideoUpload = require('@polyv/vod-upload-js-sdk/vod-upload-js-sdk.min.js');
```

快速开始

初始化上传实例

首先，创建 PlvVideoUpload 实例。

```
const videoUpload = new PlvVideoUpload({
  region: 'line1', // auto：自动选择。根据IP的地区自动选择，当IP解析不出时使用默认值。
                   // line1（默认值）：华南OSS bucket，对应ab-upload.polyv.net。
                   // line2：华北OSS bucket，对应ab-upload2.polyv.net。
  events: {
      Error: (err) => {  // 错误事件回调
          console.log(err);
      },
      UploadComplete: () => {}  // 全部上传任务完成回调
  }
});
```

调用 `updateUserData()` 设置账号授权验证信息，并每隔 3 分钟更新一次

```
// 授权验证信息3分钟内有效，当 sign 过期时需要调用该方法更新
videoUpload.updateUserData({
  userid: <userid> , // Polyv云点播账号的ID
  ptime: <timestamp> , // 时间戳
  sign: <sign> , // 是根据将secretkey和ts按照顺序拼凑起来的字符串进行MD5计算得到的值(小写)
  hash: <hash> , // 是根据将ts和writeToken按照顺序拼凑起来的字符串进行MD5计算得到的值(小写)
});
```

其中 ptime、sign 和 hash 都要从服务端获取，服务端的代码示例（PHP）如下：

```
/*
 * userid、secretkey、writeToken 都可以在「云点播管理后台 -> 设置 -> API接口」页面中找到。
 */
$userid = "your userid";
$secretkey = "your sercrety";
$writeToken = "your writeToken";

$ptime = time() * 1000;
$sign = md5($secretkey . $ptime);
$hash = md5($ptime . $writeToken);
```

添加上传文件进入上传列表

```
fileSetting = { // 文件上传相关信息设置
  title: <title>,  // 标题
  desc: <desc>,  // 描述
  cataid: <cataid>,  // 上传分类目录ID
  tag: <tag>,  // 标签
  luping: 0,  // 是否开启视频课件优化处理，对于上传录屏类视频清晰度有所优化：0为不开启，1为开启
  keepsource: 0,  // 是否源文件播放（不对视频进行编码）：0为编码，1为不编码
  state:<customMessage> //用户自定义信息，如果提交了该字段，会在服务端上传完成回调时透传返回。
};
```

调用 PlvVideoUpload 实例的 `addFile(file, events, fileSetting)` 方法，添加文件到文件列表，该方法返回一个 `UploadManager` 对象：

```
var uploadManager = videoUpload.addFile(
  file, // file 为待上传的文件对象
  {
    FileStarted: function(uploadInfo) { // 文件开始上传回调
        console.log("文件上传开始: " + uploadInfo.fileData.title);
    },
    FileProgress: function(uploadInfo) { // 文件上传过程返回上传进度信息回调
        console.log("文件上传中: " + (uploadInfo.progress * 100).toFixed(2) + '%');
    },
    FileStopped: function(uploadInfo) { // 文件暂停上传回调
        console.log("文件上传停止: " + uploadInfo.fileData.title);
    },
    FileSucceed: function(uploadInfo) { // 文件上传成功回调
        console.log("文件上传成功: " + uploadInfo.fileData.title);
        // 视频vid：uploadInfo.fileData.vid
    },
    FileFailed: function(uploadInfo) { // 文件上传失败回调
        console.log("文件上传失败: " + uploadInfo.fileData.title);
    }
  },
  fileSetting
);
```

API 文档

注： 由于业务需要，开源版本的代码和文档目前已经不再更新，仅供参考。 见源代码中的 docs 文件夹或点击此处打开。

示例代码

1、JS示例

源代码中的 demo 文件夹包含两个示例：

- dev.html & dev.js：以模块化方式引入 SDK 的示例。需要修改 build 文件夹下的 webpack.dev.config.js 文件中的账号信息，然后在本项目根目录下运行 `npm run dev` ，打开浏览器访问 `http://127.0.0.1:14002/index.html` 即可。
- index.html & index.js：以 script 标签引入 SDK 的示例。需要修改 JS 文件中的 getPolyvAuthorization 变量为有效的请求地址，才能正常使用。

2、Vue示例

1、补充示例中的userid、secretkey、writeToken即可，在保利威点播后台 设置->API接口 获取。

2、需要安装element-ui 和 md5 依赖。

```
<template>
  <div class="hello">
    <div>
      <input type="file" class="upload" @change="doUpload" ref="inputer" multiple />
      <el-button type="primary" size="small" @click="startAll">全部开始</el-button>
      <el-button type="warning" size="small" @click="pauseAll">全部暂停</el-button>
      <el-button type="danger" size="small"  @click="clearAll">全部删除</el-button>
    </div>

    <div>
      <el-table :data="tableData" border style="width: 100%">
        <el-table-column prop="id" label="ID" width="180">
        </el-table-column>
        <el-table-column prop="fileName" label="文件名" width="180">
        </el-table-column>
        <el-table-column prop="size" label="文件大小" width="180">
          <template slot-scope="scope">{{ transformSize(scope.row.size)}}</template>
        </el-table-column>
        <el-table-column prop="progress" label="进度" width="180">
          <template slot-scope="scope">
            <el-progress :text-inside="true" :stroke-width="26" :percentage="scope.row.progress"></el-progress>
          </template>

        </el-table-column>
        <el-table-column prop="progress" label="操作" width="180">
          <template slot-scope="scope">
            <el-button type="text" size="small" @click="start(scope.row.id)">开始</el-button>
            <el-button type="text" size="small" @click="stop(scope.row.id)">暂停</el-button>
            <el-button type="text" size="small" @click="remove(scope.row.id)">删除</el-button>
          </template>
        </el-table-column>
      </el-table>
    </div>
  </div>
</template>

<script>
  import md5 from 'js-md5'
  import PlvVideoUpload from '@polyv/vod-upload-js-sdk';

  export default {
    name: 'demo',
    data() {
      return {
        videoUpload: null, // 视频上传实例
        userid: '',//从点播后台查看获取
        secretkey: '',//从点播后台查看获取
        writeToken: '',//从点播后台查看获取
        ptime: '', // 当前时间戳
        tableData: [] //表格数据
      }
    },
    created() {
      this.videoUpload = new PlvVideoUpload({
        region: 'line1', // (可选)上传线路，默认line1
```

```
      events: {
        Error: (err) => { // 错误事件回调
          console.log(err);
          let errMag = `（错误代码：${err.code}）${err.message}`;
          this.$alert(errMag, '标题名称', {
            confirmButtonText: '确定',
            type: 'error',
          });
        },
        UploadComplete: () => { // 全部上传任务完成回调
          console.info('上传结束：', this.videoUpload);
          console.log(this.tableData)
          this.$message({
            message: '全部上传任务完成',
            type: 'success'
          });
        }
      }
    });
  },
  mounted() {
    this.autoUpdateUserData(null, this.videoUpload);
  },
  methods: {
    start(uploaderid) {// 单个上传
      console.log(uploaderid)
      this.videoUpload.resumeFile(uploaderid);
    },
    stop(uploaderid) {// 单个暂停
      console.log(uploaderid)
      this.videoUpload.stopFile(uploaderid);
    },
    remove(uploaderid) {// 单个删除
      console.log(uploaderid)
      this.videoUpload.removeFile(uploaderid);
      this.tableData = this.tableData.filter((item) => item.id !== uploaderid)
    },
    startAll() {// 全部上传
      if (this.videoUpload) {
        this.videoUpload.startAll();
      }
    },
    pauseAll() {// 全部暂停
      if (this.videoUpload) {
        this.videoUpload.stopAll();
      }
    },
    clearAll() {// 全部删除
      if (this.videoUpload) {
        this.videoUpload.clearAll();
        this.tableData = []
        this.$refs.inputer.value =''
      }
    },
    doUpload() {// 选择文件
      let inputDOM = this.$refs.inputer; // 通过DOM取文件数据
```

```
      console.log(inputDOM.files)
      if (inputDOM.files.length > 0) {
        inputDOM.files.forEach((file, index, arr) => {
          let fileSetting = { // 文件上传相关信息设置
            title: file.name, // 标题
            desc: 'jssdk插件上传', // 描述
            cataid: '', // 上传分类目录ID
            tag: '', // 标签
            luping: 0, // 是否开启视频课件优化处理，对于上传录屏类视频清晰度有所优化：0为不开启，1为开启
            keepsource: 0, // 是否源文件播放（不对视频进行编码）：0为编码，1为不编码
            state: '' //用户自定义信息，如果提交了该字段，会在服务端上传完成回调时透传返回。
          }
          let uploadManager = this.videoUpload.addFile(
            file, // file 为待上传的文件对象
            {
              FileStarted: this.onFileStarted,// 文件开始上传回调
              FileProgress: this.onFileProgress,// 文件上传中回调
              FileSucceed: this.onFileSucceed,// 文件上传成功回调
              FileFailed: this.onFileFailed,// 文件上传失败回调
              FileStopped: this.onFileStopped,// 文件上传停止回调
            },
            fileSetting
          );

          this.addTableData(uploadManager)
        })
      }
    },
    onFileStarted(data) {
      console.log("文件上传开始: ", data);
      this.tableData.filter((item) => item.id === data.uploaderid)[0].progress = 0
    },
    onFileProgress(data) {
      let p = parseInt(data.progress * 100);// 上传的进度条
      console.log("文件上传中: ", data);
      this.tableData.filter((item) => item.id === data.uploaderid)[0].progress = p

    },
    onFileSucceed(data) {
      console.log("文件上传成功: ", data);
      // 视频vid：data.fileData.vid
    },
    onFileFailed(data) {
      console.log("文件上传失败: ", data);
    },
    onFileStopped(data) {
      console.log("文件上传停止: ", data);
    },
    addTableData(data) { // 增加表格数据
      let obj = {
        id: data.id,
        fileName: data.fileData.title,
        size: data.fileData.size,
        progress: 0
      }
      this.tableData.push(obj)
```

```
      },
      autoUpdateUserData(timer, videoUpload) { // 启动获取用户信息
        this.getUserData(videoUpload);
        if (timer) {
          clearTimeout(timer);
          timer = null;
        }
        timer = setTimeout(() => {
          this.autoUpdateUserData(timer, videoUpload);
        }, 3 * 50 * 1000);
      },
      getUserData() { // 获取用户详细信息
        this.ptime = new Date().getTime()
        let userData = {
          userid: this.userid,
          ptime: this.ptime,
          sign: this.getSignData().sign,
          hash: this.getSignData().hash
        };
        this.videoUpload.updateUserData(userData);
      },
      getSignData() { // 加密信息参数
        let hash = md5(this.ptime + this.writeToken)
        let sign = md5(this.secretkey + this.ptime)
        return {
          hash: hash,
          sign: sign,
        }
      },
      transformSize(bytes) {// 文件大小转换
        const bt = parseInt(bytes);
        let result;
        if (bt === 0) {
          result = '0B';
        } else {
          const k = 1024;
          const sizes = ['B', 'KB', 'MB', 'GB', 'TB', 'PB', 'EB', 'ZB', 'YB'];
          const i = Math.floor(Math.log(bt) / Math.log(k));
          if (typeof i !== 'number') {
            result = '-';
          } else {
            result = (bt / Math.pow(k, i)).toFixed(2) + sizes[i];
          }
        }
        return result;
      }
    },
  }
</script>

<style scoped>
</style>
```

错误代码

Error 事件已知错误类型：

code	描述
102	用户剩余空间不足
110	文件重复
111	拦截文件类型不在 acceptedMimeType 中的文件
112	文件已经开始上传或已上传完毕，禁止修改文件信息

FileFailed 事件已知错误类型：

type	code	描述
InitUploadError	3001	分类不存在
InitUploadError	405	上传视频初始化失败
InitUploadError	406	视频大小不能为0
InitUploadError	408	账户服务状态异常，请联系客服
MultipartUploadError		断点续传时出错
UpdateTokenError		更新上传token时获取token失败
NoSuchUploadError		Multipart Upload ID 不存在

版本更新

v1.12.0

- 内部优化
- 修复频繁调用接口stopFile导致任务队列异常的情况

v1.10.1

- 优化大文件上传token更新的机制

v1.7.0

- 支持.m4a格式媒体文件上传

v1.6.0

- 修复线路授权过期重试异常的问题

v1.5.0

- 修复特定情况下断点续传异常的问题

v1.4.1

- 内部优化
- 默认线路问题修复

v1.3.0

- 上传线路升级

v1.2.3

- 增加region参数

v1.2.2

- cataid 不存在时返回提示。
- 规范FileFailed事件返回的数据格式与字段名称。

v1.2.1

- 问题修复

v1.2.0

- 支持使用子账号信息上传视频文件

v1.1.3

- 优化文件上传失败时的回调message

v1.1.2

- 增加支持文件名后缀大写的文件上传，如 file_example.MP3
- 修改示例代码

v1.1.1

- 优化文件上传失败时的重试逻辑
- 文件上传失败时返回的错误信息中增加 errData 属性

v1.1.0

- 增加对自定义信息字段的支持

v1.0.0