石山Android SDK使用说明

SmvpSDK提供的功能主要有播放视频、获取和更新视频信息、上传视频和下载视频。

一、SmvpClient的使用说明

SmvpClient是APP和SmvpSDK的交互的桥梁，所以使用SmvpSDK必须首先生成SmvpClient对象。SmvpClient的使用方法如下:

① 创建SmvpClient对象。 SmvpClient采用单例模式实现。

public static SmvpClient getInstance(Application application， String token)

token：授权 token

application: 避免内存泄露，通过 Application 获取Context

② 获取 VideoManager 对象， VideoManager类的功能是获取视频信息

public VideoManager getVideoManager()

③获取UploadManager对象， UploadManager类的功能是上传视频

public UploadManager getUploadManager()

④获取DownloadManager对象， DownloadManager类的功能是下载视频

public DownloadManager getDownloaderManager(File storageDir)

storageDir：下载视频的存储目录

⑤ APP退出时，释放资源，必须调用

public void release()

二、 ResponseListener 使用说明

从服务器获取信息是异步实现，所以需要通过ResponseListener返回结果。需要重写onSuccess和onFailure，具体使用方法如下:

① public abstract void onSuccess(String response)

response：从服务器返回的请求结果

② public abstract void onFailure(Throwable throwable)

throwable：请求失败的错误信息

三、VideoManager 的使用说明

① 获取单个视频的元数据:

public void get(String videoId， ResponseListener responseListener)

videoId：视频的id

responseListener：回调方法

获取视频信息成功后返回的数据如下，可以通过Gson将其转换为VideoData，方便使用:

{

"audio\_channels":1，

"password":null，

"id":"647026622611877784"，

"thumbnail\_url":"XX"，

"title":"update video"，

"audio\_kbps":64，

"height":600，

"description":"test update"，

"activated":true，

"user\_id":"-1"，

"video\_kbps":387，

"framerate":15，

"tags":["update1"，"update2"]，

"file\_size":7247817，

"status":"FINISHED"，

"width":800，

"modified\_time":"2014-10-13 13:42:41"，

"error\_code":null，

"category\_id":"640378515748843249"，

"audio\_sample\_rate":44100，

"extension":"。mp4"，

"snapshot\_url":"XX"，

"category":"test1"，

"duration":127460，

"categories":"640378515748843249"，

"created\_time":"2014-10-10 14:33:45"

}

② 获取多个视频的元数据:

public void list(int start， int max， String categoryId， ResponseListener responseListener)

start ：视频的起点位置

max：获取视频的最大数量

categoryId：视频的分类id，为null时，返回所有分类的视频

responseListener：回调方法

获取视频信息成功后返回的数据如下，可以通过Gson将其转换为VideoData，方便使用:

{

"total":60， 总视频数量

"items":

[{

"tags":[]，

"file\_size":709666，

"status":"FINISHED"，

"width":948，

"modified\_time":"2015-01-18 20:46:43"，

"error\_code":null，

"category\_id":"-1"，

"audio\_channels":1，

"password":null，

"audio\_sample\_rate":44100，

"extension":"。mp4"，

"id":"684154553318270345"，

"snapshot\_url":"XX"，

"thumbnail\_url":"XX"，

"title":"test2"，

"duration":11100，

"audio\_kbps":28，

"height":540，

"description":null，

"activated":true，

"user\_id":"-1"，

"categories":"-1"，

"video\_kbps":526，

"created\_time":"2015-01-18 15:49:05"，

"framerate":20

}，

……

]

}

③ 获取单个视频的转码信息

public void json (String videoId， ResponseListener responseListener)

videoId：视频Id

responseListener：回调方法

获取视频信息成功后返回的数据如下:

{

"entries":

[

{

"id":"684154553318270345"，

"subtitles":[]，

"duration":11100，

"title":"test2"，

"renditions":[

{

"file\_size":1153325，

"status":"FINISHED"，

"urls":

{

"mp4":{"url":"XX。mp4"，"expires":604800}，

"smf":{"url":"XX。smf"，"expires":604800}，

"m3u8":{"url":"XX。m3u8"，"expires":604800}，

"thumbnail":{"url":"XX。jpg"，"expires":604800}

}，

"width":948，

"error\_code":null，

"type":"ANDROID"，

"id":"684516717845571311"，

"height":540，

"default":false，

"kbps":964，

"download\_url":"XX"，

"name":"ANDROID\_HD"，

"setting\_id":"640361859865669264"，

"framerate":20

}

]，

"cuepoints":[]，

"thumbnails":[

{

"publisher\_id":"640361859865669248"，

"file\_size":2431，

"rendition\_id":null，

"width":120，

"data":"{\"width\": 120， \"group\": 5550， \"height\": 90}"，

"type":"SNAPSHOT"，

"deleted":false，

"url":"XX。jpg"，

"id":"684155609930548633"，

"entry\_id":"684154553318270345"，

"height":90，

"refer\_id":"684154553318270345"，

"file\_key":"XX。jpg"，

"group":5550，

"created\_time":"2015-01-18 15:53:11"

}，

{

"publisher\_id":"640361859865669248"，

"file\_size":24051，

"rendition\_id":null，

"width":948，

"data":"{\"width\": 948， \"group\": 5550， \"height\": 540}"，

"type":"SNAPSHOT"，

"deleted":false，

"url":"XX。jpg"，

"id":"684155609930548631"，

"entry\_id":"684154553318270345"，

"height":540，

"refer\_id":"684154553318270345"，

"file\_key":"XX。jpg"，

"group":5550，

"created\_time":"2015-01-18 15:53:11"

}

]，

"password":null

}

]，

"title":"test2"

}

④ 更新单个视频信息

方法一：

public void update(String videoId， String title， String description， String[] tags，ResponseListener responseHandler)

videoId: 视频id

title: 名称

description：描述

tags：标签

方法二：

public void update(VideoData videoData， ResponseListener responseListener)

更新成功后返回的结果如下，可以通过Gson将其转换为VideoData，方便使用:

{

"audio\_channels":1，

"password":null，

"id":"647443938814226137"，

"thumbnail\_url":"XX"，

"title":"update video"，

"audio\_kbps":64，

"height":600，

"description":"test update"，

"activated":true，

"user\_id":"-1"，

"video\_kbps":387，

"framerate":15，

"tags":["update1"，"update2"]，

"file\_size":7247817，

"status":"FINISHED"，

"width":800，

"modified\_time":"2015-01-19 16:19:15"，

"error\_code":null，

"category\_id":"640378515748843249"，

"audio\_sample\_rate":44100，

"extension":"。mp4"，

"snapshot\_url":"XX"，

"category":"test1"，

"duration":127460，

"categories":"640378515748843249"，

"created\_time":"2014-10-11 17:33:09"

}

⑤ 视频上线

public void activate(String videoId)

⑥ 视频下线

public void deactivate(String videoId)

⑦ 获取单个播放器的信息

public void getPlayer(String playerId， ResponseListener responseListener)

playerId：播放器id

可以将返回结果通过Gson转换为PlayerProperty

{"show\_external\_name": true，

"logo\_overlay": ""，

"play\_button\_overlay": "true"，

"show\_ad\_tip": "true"，

"show\_rendition\_selection": "true"，

"height": "480"，

"enable\_subtitle": "true"，

"control\_bar\_auto\_hide": "false"，

"auto\_play": "false"，

"own\_ads": []，

"auth\_api\_url": ""，

"company\_link": ""，

"width": "640"，

"muted": "false"，

"show\_menu\_bar": "true"，

"swf\_url": "XX"，

"company\_name": ""，

"show\_network\_tip": "true"，

"google\_ad": null，

"control\_bar\_mode": "docked"，

"type": 3，

"show\_advanced\_settings": "false"，

"show\_light\_button": "true"，

"show\_share\_sites": "true"，

"description": "XX"，

"ga\_tracker\_id": "UA-33166483-1"，

"enable\_memorized": "true"，

"preload\_time": "60"，

"forbid\_drag": "false"，

"show\_title": "false"，

"show\_full\_screen": "true"，

"show\_email\_share": "true"，

"name": "XX"，

"show\_player\_info": "true"，

"show\_recommendation": "true"，

"allowed\_domains": ""，

"auth\_key": ""，

"scale\_mode": "letterbox"，

"allowed\_ips": ""，

"control\_bar\_auto\_hide\_timeout": "2"，

"show\_company\_info": "true"，

"download\_enabled": "true"，

"skin": {"definition": ""， "preview": "XX"， "publisher\_id": "-1"， "name": "默认"， "id": "0"}，

"loop": "false"

}

⑧ 获取所有播放器的信息

public void getAllPlayers(ResponseListener responseListener)

可以将返回结果通过Gson转换为SimplePlayerProperty

[

{

"description": "XXX"，

"defaults": {"width": "640"， "auto\_play": "true"， "height": "480"}，

"created\_time": "2013-02-27 14:20:40"，

"preview": "XXX"，

"type": 3，

"id": "428082998184697028"，

"name": "XXXX"}，

XXXX

]

⑨APP退出的时候，释放资源，必须调用

public void release()

⑩相关的类：

VideoData、SimplePlayerProperty和PlayerProperty是简单的工具类，VideoData实现了Parcelable接口。

四、上传视频说明

4.1 UploadManager 使用说明

UploadManager类管理上传视频，主要功能有上传视频和取消上传。上传视频不支持暂停功能；APP退出时，UploadManager会自动取消所有正在上传的任务，APP再次启动后，自动开始未完成上传的任务。同时最多只能执行1个上传任务，其他任务处于等待状态。通过UploadListener获取上传状态。

① 获取UploadManager，调用SmvpClient的 getUploadManager()方法

public UploadManager getUploadManager()

②上传单个视频

public UploadTask upload(UploadData uploadData)

③批量上传

public List<UploadTask> upload(List<UploadData> uploadDataList)

④全部开始

public List<UploadTask> uploadAll()

⑤取消上传

public void cancel(UploadTask uploadTask)

⑥全部取消

public void cancelAll()

⑦ 获取所有的上传任务

public List<UploadTask> getTaskList()

4.2 UploadData使用说明

UploadData用来组织上传需要的数据，有简单上传和完整信息上传两个构造方法，上传任务在子线程执行，

① 完整信息上传

public UploadData(String title, String description, String[] tags,

String categoryId, boolean activated, File videoFile)

参数说明：

title: 视频名称

description：描述

tags：标签

categoryId：视频分类Id

activated：是否上线

videoFile：需要上传的文件

② 简单上传

public UploadData(String title, String description, File videoFile)

title: 视频名称

description：描述

videoFile：需要上传的文件

简单上传时，tags为null，categoryId为null，activated使用默认值true

③ 设置UploadListener 获取上传状态

public void setUploadListener(UploadListener uploadListener)

4.3 UploadListener 使用说明

UploadListener 用来监听上传任务的执行情况，需要实现的方法如下:

① 上传成功

void onSuccess(JSONObject response)

上传成功时返回的信息如下:

{

"audio\_channels":null，

"password":null，

"id":"685198427939693825"，

"thumbnail\_url":null，

"title":"test4"，

"audio\_kbps":null，

"height":null，

"description":"description4"，

"activated":true，

"user\_id":"-1"，

"video\_kbps":null，

"framerate":null，

"tags":["tag3"，"tag4"，"tag5"]，

"file\_size":709666，

"status":"PENDING"，

"width":null，

"modified\_time":"2015-01-21 11:19:51"，

"error\_code":null，

"category\_id":"640378515748843249"，

"audio\_sample\_rate":null，

"extension":"。mp4"，

"snapshot\_url":null，

"category":"test1"，

"duration":null，

"categories":"640378515748843249"，

"created\_time":"2015-01-21 11:19:51"

}

简单上传和完整信息上传时返回的信息“tags”，”category'和“category\_id”不同，简单上传返回的信息：

"tags":[] ; "category\_id":"-1"; "category":"uncategorized"

② 更新上传进度，progress为0到100的整数

void onProgressChanged(int progress)

③ 更新上传状态

void onStatusChanged(int status)

SDKConstants.STATUS\_PENDING：等待上传

SDKConstants.STATUS\_RUNNING：正在上传

SDKConstants.STATUS\_SUCCESS： 上传成功

SDKConstants.STATUS\_FAILURE：上传失败

SDKConstants.STATUS\_WAIT\_NETWORK：等待网络连接

④ 上传失败

void onFailure(Exception e)

4.4 UploadTask 使用说明

UploadTask继承了AsyncTask，是执行上传任务的子线程。APP可以使用的方法如下，请不要调用其他方法：

① 获取上传数据

public UploadData getUploadData()

② 是否正在上传

public boolean isRunning()

③ 获取上传进度

public int getProgress()

④ 获取已经上传的字节数

public long getTransferedBytes()

⑤ 获取总的字节数

public long getTotalBytes()

⑥ 设置 UploadListener

public void setUploadListener(UploadListener uploadListener)

⑦ 获取上传状态

public int getUploadStatus()

有以下几种状态:

SDKConstants.STATUS\_PENDING：等待上传

SDKConstants.STATUS\_RUNNING：正在上传

SDKConstants.STATUS\_SUCCESS： 上传成功

SDKConstants.STATUS\_FAILURE：上传失败

SDKConstants.STATUS\_WAIT\_NETWORK：等待网络连接

五、下载视频说明

5.1 DownloadManager使用说明

DownloadManager类管理下载视频相关的操作，主要功能有下载视频， 暂停下载和取消下载。APP退出时，DownloadManager会自动暂停所有的下载任务;APP再次启动，初始化DownloadManager时，会自动开始下载未完成的任务。下载任务在子线程执行，同时最多能运行3个下载任务，其他任务处于等待状态;通过DownloadListener获取下载状态。

① 获取DownloadManager，调用SmvpClient的 getDownloaderManager()方法

public DownloadManager getDownloaderManager(File storageDir)

storageDir: 存储路径，内部存储或者外部存储

②下载视频

public DownloadTask download(DownloadData downloadData)

③ 全部开始

public List<DownloadTask> downloadAll()

④ 暂停下载

public void stop(DownloadTask downloadTask)

⑤ 全部暂停

public void stopAll()

⑥ 取消下载

public void cancel(DownloadTask downloadTask)

⑦ 全部取消

public void cancelAll()

⑧ 获取所有的下载任务

public List<DownloadTask> getTaskList()

5.2 DownloadData使用说明

DownloadData用来组织下载需要的数据

① 构造方法说明

public DownloadData(String videoId, String definition)

videoId: 视频id

definition：清晰度

清晰度有以下几种：

SDKConstants.DEFINITION\_ANDROID\_SMOOTH：流畅

SDKConstants.DEFINITION\_ANDROID\_STANDARD：标清

SDKConstants.DEFINITION\_ANDROID\_HD：高清

② 获取下载状态

public void setDownloadListener(DownloadListener downloadListener)

5.3 DownloadListener 使用说明

DownloadListener 用来监听下载任务的执行情况，需要实现的方法如下:

① 下载成功

void onSuccess()

② 更新下载进度

void onProgressChanged(int progress);

③ 更新下载状态

void onStatusChanged(int status)

④ 下载失败

void onFailure(Exception e)

需要注意“NO\_RENDITION\_AVAILABLE”的错误，这个错误是由服务器没有对应码率的视频引起的，需要转码。

5.4 DownloadTask 使用说明

DownloadTask继承了AsyncTask，是执行下载任务的子线程。通过DownloadTask能够获取下载进度和下载状态等信息。

① 构造方法

public DownloadTask(Context context, DownloadData downloadData)

② 获取下载数据

public DownloadData getDownloadData()

③ 是否正在下载

public boolean isRunning()

④ 获取下载进度，下载的视频为m3u8和ts文件，不提供下载的字节数，只提供下载进度

public int getProgress()

⑤ 设置 DownloadListener

public void setDownloadListener(DownloadListener downloadListener)

⑥ 获取下载状态

public int getDownloadStatus()

有以下几种状态:

SDKConstants.STATUS\_PENDING：等待下载

SDKConstants.STATUS\_RUNNING：正在下载

SDKConstants.STATUS\_STOP\_BY\_USER：用户停止下载

SDKConstants.STATUS\_AUTO\_STOP：自动停止下载（APP退出时自动设定）

SDKConstants.STATUS\_SUCCESS：下载成功

SDKConstants.STATUS\_CANCELLED：取消下载

SDKConstants.STATUS\_FAILURE：下载失败

SDKConstants.STATUS\_WAIT\_NETWORK：等待网络连接

六、播放视频

6.1 VideoView使用说明

视频播放主要是由VideoView类完成，和Android的VideoView类功能类似，既可播放石山视频云服务平台下的视频（以下简称在线播放），也可播放本地视频。VideoView是自定义View，继承FrameLayout，可以像普通View一样使用。

例如：

<cn.smvp.android.sdk.view.VideoView

android:id="@+id/video\_view"

android:layout\_width="match\_parent"

android:layout\_height="250dp" />

VideoView默认是小窗口播放,类似优酷播放视频的方式,打开一个视频,默认是半个窗口大小播放,点击全屏按钮可以全屏播放,点击缩小按钮,回到小窗口播放,如果要直接全屏播放,请参考全屏播放的使用说明。

6.2 在线播放使用说明

① 播放网络视频

public void playVideo(VideoManager videoManager, String videoId, final String playerId, final String definition)

videoManager：可以通过SmvpClient获取

videoId：视频id

playerId：播放器id

definition：默认清晰度

视频的清晰度有以下几种：

SDKConstants.DEFINITION\_ANDROID\_SMOOTH：流畅

SDKConstants.DEFINITION\_ANDROID\_STANDARD：标清

SDKConstants.DEFINITION\_ANDROID\_HD：高清

②切换正在播放的视频到其他视频

public void changeVideo(String videoId, final String definition) {

videoId: 视频的id

definition: 默认清晰度

示例:

String videoId="XX"

String definition = SDKConstants.DEFINITION\_ANDROID\_STANDARD;

mVideoView.changeVideo(videoId, definition);

③ 使用VideoView,点击全屏按钮切换到全屏播放

VideoView已经实现了全屏播放的功能，用户点击全屏/缩放按钮，会切换到全屏/非全屏状态。用户切换屏幕方向时也会自动切换为全屏/非全屏播放。全屏播放时可以更改播放视屏的清晰度和调节音量。

相关方法如下：

public boolean rotatedFromBtn() 判断是否点击全屏/缩放按钮

public void toFullScreen() 全屏播放

public void toSmallScreen() 退出全屏状态

具体实现：

1. AndroidManifest中，播放视频的Activity需要添加"android:configChanges", PlayActivity为测试使用，不在SDK中。

<activity

android:name=".PlayActivity"

android:label="@string/title\_activity\_play\_video"

android:configChanges="keyboardHidden|orientation|screenSize"/>

2. 重写onConfigurationChanged()

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

try {

if (mVideoView != null) {

if (mVideoView.rotatedFromBtn()) {

return;

} else {

if (mVideoView.isFullScreen()) {

mVideoView.toMiniScreen();

} else {

mVideoView.toFullScreen();

}

}

}

} catch (Exception e) {

MyLogger.w(LOG\_TAG, "onConfigurationChanged exception");

}

}

④只有全屏播放,没有小窗口播放模式

PlayVideoActivity也可以实现全屏播放在线视频，这种情况，只有全屏窗口，不能切换到小窗口播放。使用方法如下，所有的参数都是必须的：

String videoId="XX"

String playerId="XX"

Intent intent = new Intent();

intent.putExtra(SDKConstants.KEY\_ID, videoId);

intent.putExtra(SDKConstants.KEY\_DEFINITION, SDKConstants.DEFINITION\_ANDROID\_HD);

intent.putExtra(SDKConstants.KEY\_PLAYER\_ID, playerId);

intent.putExtra(SDKConstants.KEY\_TOKEN, LocalConstants.TOKEN);

intent.putExtra(SDKConstants.KEY\_PLAY\_MODE, VideoView.PLAY\_MODE\_ONLY\_FULL);

intent.setClass(getActivity(), PlayVideoActivity.class);

startActivity(intent);

参数说明：

SDKConstants.KEY\_ID：视频id

SDKConstants.KEY\_DEFINITION: 默认的清晰度

SDKConstants.KEY\_PLAYER\_ID：播放器的id

SDKConstants.KEY\_TOKEN：token

SDKConstants.KEY\_PLAY\_MODE：播放窗口的显示模式

⑤ SDKConstants.KEY\_PLAY\_MODE 说明

PlayVideoActivity播放视频时，使用KEY\_PLAY\_MODE 区分播放在线视频和离线视频。

在线视频： VideoView.PLAY\_MODE\_ONLY\_FULL

离线视频： VideoView.PLAY\_MODE\_ONLY\_ OFFLINE

6.3 播放离线视频使用说明（指下载到手机的视频）

播放本地视频主要由PlayVideoActivity完成，APP通过给PlayVideoActivity发intentl来播放本地的视频。播放本地视频只有全屏播放，可以调节音量。PlayVideoActivity目前只支持播放本地视频。

intent传递的参数：

key: SDKConstants.KEY\_AUTO\_START value: true/false 是否自动播放，

key: SDKConstants.KEY\_VIDEO\_DIRECTORY value: 视频的存储目录，为m3u8文件的上层目录

例如：

Intent intent = new Intent();

intent.setClass(mContext, PlayVideoActivity.class);

intent.putExtra(SDKConstants.KEY\_AUTO\_START, true);

intent.putExtra(SDKConstants.KEY\_PLAY\_MODE, VideoView.PLAY\_MODE\_OFFLINE);

intent.putExtra(SDKConstants.KEY\_VIDEO\_DIRECTORY,

“/sdcard/smvpdemo/648151899748462422\_ANDROID\_HD”);

mContext.startActivity(intent);

6.4 获取当前的播放时间

播放视频的时候（包括全屏播放和离线播放），会发送当前播放时间的广播，发送的信息如下：

action: SDKConstants.ACTION\_PLAY\_PROGRESS\_CHANGED

Intent intent = new Intent(SDKConstants.ACTION\_PLAY\_PROGRESS\_CHANGED);

intent.putExtra(SDKConstants.KEY\_ID, “XXX”); 视频id

intent.putExtra(SDKConstants.KEY\_TITLE, “XXX”); 视频名称

intent.putExtra(SDKConstants.KEY\_PROGRESS, X.XXX);; 当前播放的时间，以秒为单位，最多3为小数

接收消息方法举例：

ProgressReceiver mProgressRecevier = new ProgressReceiver();

IntentFilter filter = new IntentFilter();

filter.addAction(SDKConstants.ACTION\_PLAY\_PROGRESS\_CHANGED);

registerReceiver(mProgressRecevier, filter)

private class ProgressReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (SDKConstants.ACTION\_PLAY\_PROGRESS\_CHANGED == action) {

String id = intent.getStringExtra(SDKConstants.KEY\_ID);

String title = intent.getStringExtra(SDKConstants.KEY\_TITLE);

float progress = intent.getFloatExtra(SDKConstants.KEY\_PROGRESS, 0);

}

}

}

6.5 常用方法

① public void setOnPreparedListener(IMediaPlayer.OnPreparedListener l)

②public void setOnCompletionListener(IMediaPlayer.OnCompletionListener l)

③public void setOnErrorListener(IMediaPlayer.OnErrorListener l)

④public void setOnBufferingUpdateListener(IMediaPlayer.OnBufferingUpdateListener l)

⑤ public int getVideoWidth()

⑥ public int getVideoHeight()