目錄

简介

欢拓云课堂Android SDK 是一套基于 armeabi、armv7、arm64 处理器设备的应用程序接口，提供直播点播功能，不仅支持默认官方UI，并且支持自定义UI。您可以使用欢拓云课堂Android SDK 让您的APP直接拥有观看直播、观看回放的功能。

前置条件

- 已经与欢拓网络公司达成协议
- 有获取access_token的途径
    - 获取access_token参数，详情参考:如何与我的APP进行对接

开发环境配置

- 引入SDK
  - 将libs目录下的jar文件拷贝到项目工程Application Module的libs目录
  - 右键点击jar文件，并点击弹出菜单中的“Add As Library”并将jar文件作为类库添加到项目中
  - 在项目工程Application Module的src/main目录中新建名为“jniLibs”的目录
  - 将libs/armeabi-v7a目录拷贝到“jniLibs”目录中
- 引入依赖库
  - 在Application module的build.gradle文件添加第三方依赖库

```
   dependencies {
        compile 'com.google.code.gson:gson:2.8.0'
        implementation 'com.github.bumptech.glide:glide:4.5.0'
        annotationProcessor 'com.github.bumptech.glide:compiler:4.5.0'
        compile 'com.squareup.retrofit2:retrofit:2.3.0'
        compile 'com.squareup.retrofit2:converter-gson:2.3.0'
        compile 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
        compile 'io.reactivex.rxjava2:rxandroid:2.0.1'
        compile 'io.reactivex.rxjava2:rxjava:2.1.0'
        compile 'io.socket:socket.io-client:0.8.3'
        compile 'org.nanohttpd:nanohttpd:2.2.0'
         }
```

  - 配置AndroidManifest.xml
  - 添加SDK需要的权限到标签下

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

- 在6.0之后，需要动态请求权限：需要动态请求的有：

```
String[] PERMISSIONS = {Manifest.permission.WRITE_EXTERNAL_STORAGE, Manifest.permissi
on.READ_EXTERNAL_STORAGE,
      Manifest.permission.CAMERA, Manifest.permission.MODIFY_AUDIO_SETTINGS, Manifest
.permission.RECORD_AUDIO};
```

  - 在相应的Activity标签添加声明configChanges属性,添加configChanges属性配

置"keyboardHidden|orientation|screenSize"匹配横屏事件，使在屏幕方向改变时系统不重启Activity

```
android:configChanges="keyboardHidden|orientation|screenSize"
```

- 使用欢拓云课堂SDK，必须传入一个access_token，该access_token的获取，参考申请密钥流程

到此，你已完成了**SDK**的依赖添加

初始化SDK逻辑

初始化

- SDK相关播放逻辑是通过HtSdk对象实现，HtSdk是一个单例对象，调用初始化方法进行初始化
- 特别说明的是，在初始化HtSdk对象时需要access_token
- 需要提供摄像头视频容器和白板容器

```
//1.首先获取容器对象和access_token值
//白板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 白板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
mHtSdk.init(context, pptContainer, videoViewContainer, access_token, TFMode.LIVE_NORMAL);

//4.设置进入后台是否暂停（默认是暂停）(可选)
mHtSdk.setPauseInBackground(true);

//5.调用onResume方法
//SDK在调用HtSdk对象的onResume方法时去加载数据
//如在Activity中调用，可对应Activity的onResume方法
//否则在初始化完成后调用
mHtSdk.onResume();
```

暂停Activity

```
//在app退回后台时，调用onPause方法
//可对应Activity的onPause方法
//如果不设置setPauseInBackground或设置为true时，执行暂停播放
mHtSdk.onPause();
```

释放注销对象

```
//如果退出播放Activity，调用release方法释放HtSdk对象
//调用release方法后，须再次调用init方法初始化

mHtSdk.release();
```

到此为止已经简单地实例化了**SDK**，已经具有播放音视频以及显示**ppt**的功能。

初始化SDK逻辑

链接：

- RTC接口调用
- 涂鸦
- 直播时长

初始化

- SDK相关播放逻辑是通过HtSdk对象实现，HtSdk是一个单例对象，调用初始化方法进行初始化 *特别说明的是，在初始化HtSdk对象时需要access_token *需要提供摄像头视频容器和白板容器

```
//1.首先获取容器对象和access_token值
//白板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 白板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
//context 为上下文
mHtSdk.init(context,pptContainer, videoViewContainer, access_token，TFMode.LIVE_MIX);

//4.设置进入后台是否暂停（默认是暂停）(可选)
mHtSdk.setPauseInBackground(true);

//5.调用onResume方法
//SDK在调用HtSdk对象的onResum方法时去加载数据
//如在Activity中调用，可对应Activity的onResume方法
//否则在初始化完成后调用
mHtSdk.onResume();
```

暂停Activity

```
//在app退回后台时，调用onPause方法
//可对应Activity的onPause方法
//如果不设置setPauseInBackground或设置为true时，执行暂停播放
mHtSdk.onPause();
```

释放注销对象

```
//如果退出播放Activity，调用release方法释放HtSdk对象
//调用release方法后，须再次调用init方法初始化

mHtSdk.release();
```

到此为止已经简单地实例化了**SDK**，已经具有播放音视频以及显示**ppt**的功能。

初始化SDK逻辑

链接：

- RTC接口调用
- 涂鸦
- 直播时长

初始化

- SDK相关播放逻辑是通过HtSdk对象实现，HtSdk是一个单例对象，调用初始化方法进行初始化 *特别说明的是，在初始化HtSdk对象时需要access_token *相较于大班的模式化，只需要提供白板容器

```
//1.首先获取容器对象和access_token值
//白板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 白板布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
//context 为上下文
mHtSdk.init(context,pptContainer, null, mToken, TFMode.LIVE_RTC);
//5.调用onResume方法
//SDK在调用HtSdk对象的onResum方法时去加载数据
//如在Activity中调用，可对应Activity的onResume方法
//否则在初始化完成后调用
mHtSdk.onResume();
```

视频回调监听

```
//需要调用setRtcMemberListener（），继承OnRtcMemberListener接口来获取上讲台用户的数据和视频。

mHtSdk.setRtcMemberListener(new OnRtcMemberListener() {
        /**
         * 用户被踢下讲台
         * @param upUserEntity
         */
        @Override
        public void onKick(UpUserEntity upUserEntity) {
        }

        /**
         * 用户上讲台（包含主播）
         *
         * @param upUserEntity 用户数据
         * @param videoView 视频view
         */
        @Override
        public void onUp(UpUserEntity upUserEntity, View videoView) {
        }

        /**
         * 用户下讲台
         * @param upUserEntity
         */
        @Override
        public void onDown(UpUserEntity upUserEntity) {

        }

        /**
         *用戶离线
         * @param upUserEntity
         * @param reason
         */
        @Override
        public void onOffline(UpUserEntity upUserEntity, int reason) {}

    });
```

暂停Activity

```
//在app退回后台时，调用onPause方法
//可对应Activity的onPause方法
mHtSdk.onPause();
```

释放注销对象

```
//如果退出播放Activity，调用release方法释放HtSdk对象
//调用release方法后，须再次调用init方法初始化

mHtSdk.release();
```

到此为止已经简单地实例化了**SDK**，已经具有播放音视频以及显示**ppt**的功能。

初始化生活直播SDK逻辑

初始化

- 生活直播SDK相关播放逻辑是通过HtLifeLiveSdk对象实现，HtLifeLiveSdk是一个单例对象，调用初始化方法进行初始化
- 特别说明的是，在初始化HtLifeLiveSdk对象时需要access_token
- 需要提供摄像头视频容器

```
//1.首先获取容器对象和access_token值
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtLifeLiveSdk mSdk = HtLifeLiveSdk.getInstance();

//3.创建InitParams传入初始化参数并调用HtLifeLiveSdk#init()方法进行初始化
InitParams params = new InitParams();
params.token = mToken;
params.videoViewContainer = mBinding.videoContainer;
Context context = getApplication();
mSdk.init(context,params);


//4.调用onResume方法
//SDK在调用HtLifeLiveSdk对象的onResume方法时去加载数据
//如在Activity中调用，可对应Activity的onResume方法
//否则在初始化完成后或在合适的时机调用
mSdk.onResume();
```

暂停Activity

```
//在app退回后台时，调用onPause方法
//可对应Activity的onPause方法或在合适的时机调用，执行暂停播放
mSdk.onPause();
```

释放注销对象

```
//如果退出播放Activity，调用release方法释放HtSdk对象
//调用release方法后，须再次调用init方法初始化
mSdk.release();
```

到此为止已经简单地实例化了**SDK**，已经具有播放音视频的功能。

初始化SDK逻辑

创建对象

- SDK相关播放逻辑是通过HtSdk对象实现，HtSdk是一个单例对象，调用初始化方法进行初始化
- 特别说明的是，在初始化HtSdk对象时需要access_token
- 需要提供摄像头视频容器和ppt容器

```
//1.首先获取容器对象和access_token值
//白板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 白板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
//调用init重载方法
mHtSdk.init(context, pptContainer, videoViewContainer, access_token, TFMode.PLAYBACK_NORM
AL);


//4.设置进入后台是否暂停（默认是暂停）(可选)
mHtSdk.setPauseInBackground(true);

//5.调用onResume方法
//SDK在调用HtSdk对象的onResum方法时去加载数据
//如在Activity中调用，可对应Activity的onResume方法
//否则在初始化完成后调用
mHtSdk.onResume();
```

暂停Activity

```
//在app退回后台时，调用onPause方法
//可对应Activity的onPause方法
//如果不设置setPauseInBackground或设置为true时，执行暂停播放
mHtSdk.onPause();
```

释放注销对象

```
//如果退出播放Activity，调用release方法释放HtSdk对象
//调用release方法后，须再次调用init方法初始化
mHtSdk.release();
```

到此为止已经简单地实例化了**SDK**，已经具有播放音视频以及显示**ppt**的功能。

功能说明

修订时间： 2020-05-13 10:23:22

HtLifeLiveSdk

功能

生活直播SDK接口类

介绍

- 主要负责生活直播相关功能的封装，除直播视频基础功能外，对外提供生活直播特有的交互功能接口
- 该接口类是一个单例

- ## 接口简介

SDK基础方法

API	描述
getInstance	静态方法，获取HtLifeLiveSdk单例
init	初始化
isInited	判断sdk是否已初始化完成
onPause	暂时播放
onResume	开始或重新加载
onStop	stop方法
release	释放注销对象
reload	刷新重新加载

播放配置接口

API	描述
getCourseInfo	获取课程信息
getInitLiveStatus	获取初始化直播状态
getNetworkList	异步获取直播网络线路列表
getRoomInfo	获取房间信息
setVideoScaleMode	设置摄像头视频显示模式
setVideoViewContainer	设置视频容器
setWhiteboardViewContainer	设置白板容器
setDesktopVideoContainer	设置视频分享/插播容器
setWarmUpVideoContainer	设置暖场容器
setNetwork	设置直播网络线路
sendFlower	发送鲜花
emit	发送消息
off	注销监听指令
on	指令监听

生活直播配置接口

API	描述
getLifeConfig	获取生活直播配置信息
setInvitationListPageSize	设置邀请榜单分页条数
report	举报
sendLike	发送点赞
getFirstPageInvitationList	获取邀请榜单首页列表
getNextPageInvitationList	获取邀请榜单下页列表

事件接口

API	描述
setLiveListener	设置直播事件监听
setHtDispatchChatMessageListener	设置获取聊天信息事件监听
setHtDispatchFlowerListener	设置鲜花信息事件监听
setHtDispatchNoticeListener	设置获取公告事件监听
setHtDispatchQuestionListener	设置问答信息事件监听
setHtDispatchRollAnnounceListener	设置获取滚动通知事件监听
setHtDispatchRoomMemberNumListener	设置观看人数事件监听
setHtLotteryListener	设置抽奖事件监听
setOnMemberJoinListener	调用房间成员用户进入直播监听
setOnVideoChangeListener	设置视频切换事件监听
setOnLikeListener	设置点赞监听

- 接口详细
  - **emit**

    发送消息

    ```
    public void emit(String cmd,JSONObject data,Callback callback)
    参数:
        `cmd` - 发送消息类型
        `data` - 发送消息内容
        `callback` - 发送回调
    ```

    - **emit**

    发送消息

    ```
    public void emit(String cmd,String msg, Callback callback)
    参数:
        `cmd` - 发送消息类型
        `msg` - 消息内容
        `callback` - 发送回调
    ```

    - **getCourseInfo**

      获取课程信息

    ```
    public CourseInfo getCourseInfo()
    ```

    - **getFirstPageInvitationList**

      获取邀请榜单首页列表

public void getFirstPageInvitationList(Callback> callback)

- **getInitLiveStatus**

  获取初始化直播状态

```
public String getInitLiveStatus()
```

  - **getInstance**

    获取HtLifeLiveSdk实例

```
public static HtLifeLiveSdk getInstance()
```

  - **getLifeConfig**

    获取生活直播配置信息

public LifeConfig getLifeConfig()

- **getNetworkList**

  异步获取直播网络线路列表

```
public void getNetworkList(OnGetNetworkChoicesCallback callback
参数:
    `callback` - 回调
```

  - **getNextPageInvitationList**

    获取邀请榜单下页列表

```
public void getNextPageInvitationList(Callback<List<InvitationItem>> callback)
```

  - **getRoomInfo**

    获取房间信息

```
public com.talkfun.sdk.module.RoomInfo getRoomInfo()
```

  - **init**

    初始化

    public void init(Context context,InitParams params) 参数: `context` - 上下文 `params` - 初始化参数

  - **isInited**

    sdk是否已初始化完成

public boolean isInited()

- **off**

  注销注册所有事件监听

  public void off()

- **off**

  注销监听指令

  ```
  public void off(java.lang.String cmd)
  ```

  参数:

  `cmd` - 指令

- **off**

  注销监听指令

  public void off(String cmd,Emitter.Listener listener) 参数: `cmd` - 指令 `listener` - 事件监听

- **on**

  指令监听

  ```
  public void on(String cmd,Emitter.Listener listener)
  参数:
      `cmd` - 指令
      `listener` - 回调监听
  ```

  - **onPause**

    暂停播放 可对应Activity#onPause()方法或在合适的时机调用，执行暂停播放

  ```
  public void onPause()
  ```

  - **onResume**

    开始或重新加载

  - 调用该方法时去加载数据或重新加载数据
  - 如在Activity中调用，可对应Activity#onResume()方法
  - 否则在初始化完成后或在合适的时机调用

    public void onResume()

- **onStop**

对应Activity#onStop()方法，sdk暂无实现，暂停播放操作在onPause()方法

```
public void onStop()
```

- **release**

  释放注销对象

```
public void release()
```

- **reload**

  刷新重新加载

```
  public void reload()
```

- **report**

  举报

```
  public void report(String content,Callback callback)
  参数:
      `content` -  举报内容
      `callback` - 回调
```

- **sendFlower**

  发送鲜花

```
  public void sendFlower()
```

- **sendLike**

  发送点赞

```
  public void sendLike(int count,Callback<Integer> callback)
  参数:
      `count` - 点赞数
      `callback` - 回调
```

- **setHtDispatchChatMessageListener**

  设置获取聊天信息事件监听

  public void setHtDispatchChatMessageListener(HtDispatchChatMessageListener listener) 参数: `listener` - 聊天信息监听回调

- **setHtDispatchFlowerListener**

设置鲜花信息事件监听

public void setHtDispatchFlowerListener(HtDispatchFlowerListener listener) 参数: `listener` - 鲜花信息监听

- **setHtDispatchNoticeListener**

  设置获取公告事件监听

  public void setHtDispatchNoticeListener(HtDispatchNoticeListener noticeListener) 参数: `noticeListener` -公告事件监听

- **setHtDispatchQuestionListener**

  设置问答信息事件监听

  public void setHtDispatchQuestionListener(com.talkfun.sdk.event.HtDispatchQuestionListener questionListener) 参数: `questionListener` - 问答信息监听

- **setHtDispatchRollAnnounceListener**

  设置获取滚动通知事件监听

  public void setHtDispatchRollAnnounceListener(HtDispatchRollAnnounceListener rollAnnounceListener) 参数: `rollAnnounceListener` - 滚动通知监听

- **setHtDispatchRollAnnounceListener**

  设置观看人数事件监听

  public void setHtDispatchRoomMemberNumListener(com.talkfun.sdk.event.HtDispatchRoomMemberNumListener memberNumListener) 参数: `memberNumListener` - 观看人数事件监听

- **setHtLotteryListener**

  设置抽奖事件监听

  public void setHtLotteryListener(HtLotteryListener listener) 参数: `listener` - 抽奖事件监听

- **setInvitationListPageSize**

  设置邀请榜单分页条数

  public void setInvitationListPageSize(int size) 参数: `size` - 分页条数

- **setLiveListener**

  设置直播事件监听

  public void setLiveListener(LiveInListener listener) 参数: `listener` - 直播事件监听

- **setNetwork**

设置直播网络线路

```
public void setNetwork(int linePosition,NetItem item,OnSetNetworkCallback callback
)
参数:
    `linePosition` - 线路索引
    `item` - 线路子项
   `callback` - 回调
```

- **setOnLikeListener**

  设置点赞监听

  ```
  public void setOnLikeListener(OnLikeListener listener)
  参数:
      `listener` -点赞监听
  ```

- **setOnMemberJoinListener**

  调用房间成员用户进入直播监听

  public void setOnMemberJoinListener(OnMemberJoinListener listener) 参数: `listener` - 用户进入直播监听

- **setOnVideoChangeListener**

  设置视频切换事件监听

  public void setOnVideoChangeListener(OnVideoChangeListener listener) 参数: `listener` -视频切换事件监听

- **setVideoScaleMode**

  设置摄像头视频显示模式

  ```
  public void setVideoScaleMode(int mode)
  参数:
      `mode` - VideoScaleMode.ASPECT\_RATIO\_ORIGIN 对比例拉伸
               VideoScaleMode.FILL\_PARENT 铺满拉伸
  ```

- **setVideoViewContainer**

  设置视频容器

  ```
  public void setVideoViewContainer(ViewGroup container)
  参数:
      `container` - 视频容器
  ```

- **setWhiteboardViewContainer**

  设置白板容器

  ```
  public void setWhiteboardViewContainer(android.view.ViewGroup container)
  参数:
      `container` - 白板容器
  ```

- **setDesktopVideoContainer**

  设置视频分享/插播容器

  ```
  public void setDesktopVideoContainer(android.view.ViewGroup container)
  参数:
      `container` - 桌面分享/插播容器
  ```

- **setWarmUpVideoContainer**

  设置暖场视频容器

  ```
  public void setWarmUpVideoContainer(android.view.ViewGroup container)
  参数:
      `container` - 暖场视频容器
  ```

生活直播配置信息

LifeConfig

功能

生活直播配置信息

介绍

- 直播初始化时从服务端获取生活直播配置信息
- 在直播事件监听回调onLaunch()后获取LifeConfig配置

`@参数说明：`

参数	类型	描述
global	GlobalBean	全局配置
goods	List	商品列表
content	ContentBean	菜单内容
adList	List	浮窗广告列表
popUps	List	弹窗广告

LifeConfig.GlobalBean

全局配置

`@参数说明：`

参数	类型	描述
switchX	SwitchBean	配置开关

LifeConfig.GlobalBean.SwitchBean 全局配置开关

`@参数说明：`

参数	类型	描述
focus	FocusBean	关注按钮开关
number	NumberBean	观看人数开关
invitationList	InvitationListBean	邀请榜单开关
countDown	CountDownBean	直播倒计时开关
store	StoreBean	商城配置
reward	RewardBean	打赏主播开关
like	LikeBean	点赞开关
more	MoreBean	更多入口开关
menu	MenuBean	菜单入口开关
background	BackgroundBean	直播间背景图
question	QuestionBean	提问开关

公共定义说明

版本：

```
HtSdk.VERSION
```

com.talkfun.sdk.consts.BroadcastCmdType 广播监听命令类型

```
static final String CHAT_SEND = "chat:send"
聊天

static final String CHAT_DISABLE = "chat:disable"
禁言

static final String CHAT_ENABLE = "chat:enable"
解除禁言

static final String CHAT_DISABLE_ALL = "chat:disable:all"
全体禁言

static final String QUESTION_ASK = "question:ask"
提问

static final String QUESTION_REPLY = "question:reply"
回复

static final String QUESTION_LIST = "question:list"
问答列表

static final String MEMBER_TOTAL = "member:total";
总人数
static final String MEMBER_FORCEOUT = "member:forceout";
用户被挤下线

static final String MEMBER_KICK = "member:kick";
被踢出房间

static final String ANNOUNCE_NOTICE = "announce:notice"
公告

static final String ANNOUNCE_ROLL = "announce:roll"
滚动通知

static final String LOTTERY_START = "lottery:start"
开始抽奖

static final String LOTTERY_STOP = "lottery:stop"
结束抽奖

static final String VOTE_NEW = "vote:new";
```

```
发起新投票

static final String VOTE_PUB = "vote:pub";
发布投票结果

static final String BROADCAST = "broadcast";
广播

static final String SIGN_NEW = "sign:new";
点名开始

static final String SIGN_END = "sign:end";
点名结束

static final String SIGN_IN = "sign:in"
签到
```

com.talkfun.sdk.consts.MemberRole 成员角色

```
 //超级管理员（主播）
    public  static final String MEMBER_ROLE_SUPER_ADMIN = "spadmin";

    //普通管理员 （助播）
    public  static final String MEMBER_ROLE_ADMIN = "admin";

    //普通用户
    public  static final String MEMBER_ROLE_USER = "user";

    //游客
    public static final String MEMBER_ROLE_GUEST = "guest";
```

com.talkfun.sdk.module.VideoModeType

视频模式类型

```
  /**摄像头视频类型*/
    public static int CAMERA_MODE = 0;
    /**桌面分享视频类型*/
    public static int DESKTOP_MODE = 2;
    /**
     * RTC 视频类型
     */
    public static int RTC_MODE = 3;
```

com.talkfun.sdk.consts.LiveStatus

直播状态

```
public static final String WAIT = "wait";//直播未开始
public static final String START = "start";//直播已开始
public static final String STOP = "stop";//直播已结束
```

com.talkfun.sdk.consts.TFMode

模式选择

```
public enum TFMode {
    /**
     * 直播默认模式
     */
    LIVE_NORMAL,
    /**
     * 直播混合模式
     */
    LIVE_MIX,
    /**
     * 直播小班模式
     */
    LIVE_RTC,
    /**
     * 点播默认模式
     */
    PLAYBACK_NORMAL
}
```

直播接口说明

修订时间： 2020-05-13 10:23:22

房间

获取房间信息

```
RoomInfo roomInfo = HtSdk.getInstance().getRoomInfo()
```

com.talkfun.sdk.module.RoomInfo 房间信息

`@参数说明：`

参数	类型	描述
action	String	直播状态
roomid	String	房间ID
user	User	用户信息
zhuBo	ZhuBo	主播信息
liveTitle	String	直播标题
noticeEntity	NoticeEntity	用户信息
rollEntity	RollEntity	用户信息
disableall	int	全体禁言状态
startTime	long	直播起始时间

com.talkfun.sdk.module.User 用户信息

`@参数说明：`

参数	类型	描述
xid	String	欢拓用户ID
uid	String	合作方用户ID
avater	String	头像地址
nickname	String	用户昵称

com.talkfun.sdk.module.ZhuBo 主播信息

`@参数说明：`

参数	类型	描述
nickname	String	主播昵称
intro	String	主播简介
flower	int	鲜花数
avater	String	头像地址
p40	String	40x40尺寸头像
p150	String	150x150尺寸头像

com.talkfun.sdk.module.NoticeEntity 公告

`@参数说明：`

参数	类型	描述
content	String	内容
time	String	公告发布时间

com.talkfun.sdk.module.RollEntity

滚动通知

`@参数说明：`

参数	类型	描述
content	String	内容
time	String	时间
duration	int	时长
link	String	链接地址

直播

直播事件监听

```
调用示例：
HtSdk.getInstance().setLiveListener(new LiveInListener() {
    @Override
    public void onLaunch() {
    //    初始化完成
    }

    @Override
    public void onInitFail(String msg) {
    //    初始化失败
    }

    @Override
    public void onLiveStart() {
    //直播开始
    }

    @Override
    public void onLiveStop() {
    //直播结束
    }

    @Override
    public void memberForceout() {
    //被挤下线
    }

    @Override
    public void memberKick() {
    //被踢出房间
    }
});
```

获取直播状态

- 在直播初始化完成回调之后可获取该直播状态

```
HtSdk.getInstance().getInitLiveStatus();
```

com.talkfun.sdk.consts.LiveStatus

直播状态

`@参数说明：`

参数	类型	描述
WAIT	static String	直播未开始
START	static String	直播已开始
STOP	static String	直播已结束

视频

配置摄像头视频

- 只需要HtSdk初始化时传入一个摄像头视频容器即可
- 课件模式播放摄像头视频时，SDK会生成摄像头视频播放器加到该容器

```
调用示例：

//1.首先获取容器对象和access_token值
//画板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 画板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
mHtSdk.init(pptContainer, videoViewContainer, access_token);

//4.（非必需）调用setWhiteboardViewContainer方法设置画板容器
FrameLayout pptContainer2;
//mHtSdk.setWhiteboardViewContainer(pptContainer2);
```

配置桌面分享/插播视频

- 初始化之后调用HtSdk的setDesktopVideoContainer()方法设置桌面分享/插播视频容器
- 如果没调用setDesktopVideoContainer()方法设置容器，默认使用画板容器，桌面分享的视频会添加到画板的上一层

```
调用示例：

HtSdk mHtSdk = HtSdk.getInstance();
mHtSdk.setDesktopVideoContainer(desktopVideoContainer);
```

配置暖场视频

- 初始化之后调用HtSdk的setWarmUpVideoContainer()方法设置暖场视频容器

```
调用示例：

HtSdk mHtSdk = HtSdk.getInstance();
mHtSdk.setWarmUpVideoContainer(warmUpVideoContainer);
```

视频监听

- 当由课件模式或桌面分享/视频插播模式切换时，会调用该事件监听的onVideoModeChanging和onVideoModeChanged方法
- 当视频（摄像头视频和桌面分享/插播视频）开始、停止播放时调用onVideoStart和onVideoStop方法
- 当摄像头视频显示隐藏时调用onCameraShow和onCamerahide方法

com.talkfun.sdk.module.VideoModeType

视频模式类型

@参数说明：

参数	类型	描述
CAMERA_MODE	int	摄像头视频类型（静态变量）
DESKTOP_MODE	int	桌面分享视频类型（静态变量）
RTC_MODE	int	RTC 视频类型（静态变量）

调用示例：

```
HtSdk.getInstance().setOnVideoChangeListener(new OnVideoChangeListener() {
    @Override
    public void onVideoStart(int mode) {
    //视频开始播放
    }

    @Override
    public void onVideoStop(int mode) {
    //视频停止播放
    }

    @Override
    public void onVideoModeChanging(int beforeMode, int currentMode) {
    //视频模式切换中
    }

    @Override
    public void onVideoModeChanged() {
    //视频模式切换完成
    }

    @Override
    public void onCameraShow() {
    //摄像头显示
    }

    @Override
    public void onCameraHide() {
    //摄像头隐藏
    }
});
```

视频音量设置

初始化之后调用HtSdk的setPlayVolume(float volume)方法设置视频音量

调用示例：

```
HtSdk.getInstance().setPlayVolume(1.0f);
```

画板

配置画板

- 只需要HtSdk初始化时传入一个画板容器即可
- 播放时SDK会生成一个画板视图加到该画板容器
- 也可在初始化之后调用HtSdk的setWhiteboardViewContainer方法设置切换画板容器

```
调用示例：

//1.首先获取容器对象和access_token值
//画板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 画板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
mHtSdk.init(pptContainer, videoViewContainer, access_token);

//4.（非必需）调用setWhiteboardViewContainer方法设置画板容器
FrameLayout pptContainer2;
//mHtSdk.setWhiteboardViewContainer(pptContainer2);
```

画板布局宽高监听

```
mHtSdk.setWhiteboardPageFrameListener(new OnWhiteboardPageFrameListener() {
            @Override
            public void onWhiteboardPageFrame(Rect rect) {

            }
        });
```

用户

新用户进入直播间: `BroadCastCmdType.MEMBER_JOIN_OTHER`

```
{"noxid":["22872173"],"cmd":"member:join:other","args":{"member":{"xid":"22872173","uid":"u_46162857","nickname":"哈哈","role":"user","gender":0,"avatar":"","a":0,"voice":{"enable":1,"grant":0},"chat":{"enable":1,"grant":0},"gid":0,"term":1},"total":1}}
```

`@参数说明：`

member

参数	类型	描述
xid	String	欢拓用户ID
uid	String	合作方用户ID
nickname	String	用户昵称
role	String	用户角色
gender	String	用户性别

Total:用户总数

```
调用示例：
HtSdk.getInstance().on(BroadcastCmdType.MEMBER_JOIN_OTHER, new Emitter.Listener() {
@Override
public void call(Object... args) {

    ...
}
```

在线用户总数: `BroadCastCmdType.MEMBER_TOTAL`

```
{"total":2}
```

`@参数说明:`

参数	类型	描述
total	int	在线人数

```
调用示例：
HtSdk.getInstance().on(BroadcastCmdType.MEMBER_TOTAL, new Emitter.Listener() {
@Override
public void call(Object... args) {
    ...
}
```

用户离开房间: `BroadCastCmdType.MEMBER_LEAVE`

```
{"noxid":[22872418],"cmd":"member:leave","args":{"xid":22872418,"uid":"u_12771357","nickn
ame":"解解渴","total":1}}
```

`@参数说明:`

参数	类型	描述
xid	String	欢拓用户ID
uid	String	合作方用户ID
nickname	String	用户昵称
total	int	剩余人数

```
调用示例：
HtSdk.getInstance().on(BroadcastCmdType.MEMBER_LEAVE, new Emitter.Listener() {
@Override
public void call(Object... args) {
    ...
}
```

踢人: `BroadCastCmdType.MEMBER_KICK`

```
{"cmd":"member:kick","args":{"xid":22873790,"nickname":"哈哈"}}
```

`@参数说明：`

参数	类型	描述
xid	String	欢拓用户ID
nickname	String	用户名

```
调用示例：
HtSdk.getInstance().on(BroadcastCmdType.MEMBER_KICK, new Emitter.Listener() {
@Override
public void call(Object... args) {
    ...
}
```

被人强迫下线: `BroadCastCmdType.MEMBER_FORCEOUT`

```
{"cmd":"member:forceout","args":{"uid":"u_90243230","xid":"22874173","role":"user","regti
me":1513851238,"state":1,"nickname":"哈哈","gender":0,"avatar":"","a":0,"sid":"ldzttx4nI3x
NNI0KAwiY","sessionid":"","serverID":"14.29.84.210:1080","gid":0,"term":1,"ip":"219.136.2
04.230","voice":{"enable":1,"grant":0},"video":{"enable":1,"grant":0},"chat":{"enable":1,
"grant":0},"heatbeatTime":1513851486,"lastTime":1513851431}}
```

`@参数说明：`

参数	类型	描述
xid	String	欢拓用户ID
uid	String	合作方用户ID
role	String	角色
nickname	String	用户名
gender	int	用户性别
state	int	状态
heatbeatTime	init	被强迫下线时间
lastTime	init	进入房间时间

```
调用示例：
HtSdk.getInstance().on(BroadcastCmdType.MEMBER_FORCEOUT, new Emitter.Listener() {
@Override
public void call(Object... args) {
    ...
}
```

聊天

sendChatMessage

发送聊天消息

```
    void sendChatMessage(String msg, Callback<ChatEntity> callback)
```

@参数说明：

参数	类型	描述
msg	String	消息内容
callback	Callback	发送消息回调

调用示例：

```
 mHtSdk.sendChatMessage("hello world", new Callback<ChatEntity>() {
                @Override
                public void success(ChatEntity result) {
                   ...
                }

                @Override
                public void failed(String failed) {
                   ...
                }
            });
```

sendChatPrivate

发送聊天消息

```
    void sendChatPrivate(int toXid, String msg, Callback<ChatEntity> callback)
```

@参数说明：

参数	类型	描述
toXid	int	私聊目标xid
msg	String	消息内容
callback	Callback	发送消息回调

```
调用示例：

 mHtSdk.sendChatPrivate(302831,"hello world", new Callback<ChatEntity>() {
                @Override
                public void success(ChatEntity result) {
                   ...
                }

                @Override
                public void failed(String failed) {
                   ...
                }
            });
```

setHtDispatchChatMessageListener

设置接收聊天信息回调接口

```
    void setHtDispatchChatMessageListener(HtDispatchChatMessageListener chatMessageListen
er)
```

```
    调用示例：
    mHtSdk.setHtDispatchChatMessageListener(new HtDispatchChatMessageListener() {
    @Override
        public void receiveChatMessage(ChatEntity entity) {
            ...
        }
    });
```

聊天信息:

ChatEntity

聊天信息

`@参数说明：`

参数	类型	描述
xid	String	用户唯一ID
uid	String	合作方用户ID
nickname	String	用户昵称
role	String	用户角色
gender	String	用户性别
avatar	String	用户头像地址
msg	String	消息内容
time	String	时间戳
attr	JSONObject	自定义扩展信息

监听聊天事件:

BroadcastCmdType.CHAT_DISABLE

监听个人禁言事件

```
    void on(BroadcastCmdType.CHAT_DISABLE,  new Emitter.Listener(){})
```

@返回数据:

```
{"cmd":"chat:disable","args":{"xid":337861,"nickname":"rrrr"}}
```

@参数说明:

参数	类型	描述
xid	String	用户唯一ID
nickname	String	用户昵称

```
调用示例：
mHtSdk.on(BroadcastCmdType.CHAT_ENABLE, new Emitter.Listener() {
    @Override
    public void call(Object... objects) {
        ...
    }
});
```

BroadcastCmdType.CHAT_ENABLE 监听解除个人禁言

```
    void on(BroadcastCmdType.CHAT_ENABLE,  new Emitter.Listener(){})
```

@返回数据:

```
{"cmd":"chat:enable","args":{"xid":337861,"nickname":"rrrr"}}
```

@参数说明:

参数	类型	描述
xid	String	用户唯一ID
nickname	String	用户昵称

```
调用示例：
mHtSdk.on(BroadcastCmdType.CHAT_ENABLE, new Emitter.Listener() {
    @Override
    public void call(Object... objects) {
        ...
    }
});
```

BroadcastCmdType.CHAT_DISABLE_ALL 监听全体禁言/解禁事件

```
    void on(BroadcastCmdType.CHAT_DISABLE_ALL,  new Emitter.Listener(){})
```

@返回数据:

```
{"cmd":"chat:chat:disable:all","args":{"status":0}}
```

@参数说明:

参数	类型	描述
status	int	禁言状态(0:解禁，1:禁言)

```
调用示例：
mHtSdk.on(BroadcastCmdType.CHAT_DISABLE_ALL, new Emitter.Listener() {
    @Override
    public void call(Object... objects) {
        ...
    }
});
```

提问

调用接口

提问: `BroadcastCmdType.QUESTION_ASK`

`@参数说明：`

参数	类型	描述
msg	String	提问内容
callback	Callback	发送回调

```
调用示例：
 mHtSdk.emit(BroadcastCmdType.QUESTION_ASK, "hello world", new Callback() {
                @Override
                public void success(Object result) {
                   ...
                }

                @Override
                public void failed(String failed) {
                   ...
                }
            });
```

回复提问: `BroadcastCmdType.QUESTION_REPLY`

`@参数格式:`

参数	类型	描述
replyId	String	回复的提问ID
msg	String	回复内容
callback	Callback	发送回调

```
调用示例：
    JSONObject data = new JSONObject();
    data.put("replyId", "40437");
    data.put("msg","老师我有个问题!")
    mHtSdk.emit(BroadcastCmdType.QUESTION_REPLY, data, new Callback() {
        @Override
        public void success(Object result) {
            ...
        }
        @Override
        public void failed(String failed) {
            ...
        }
    }
```

广播通知

接收问答:

com.talkfun.sdk.module.QuestionEntity

问答信息

`@参数说明：`

参数	类型	描述
id	String	问答ID
xid	String	用户唯一ID
uid	String	合作方用户ID
replyId	String	回复的问题ID
role	String	角色
content	String	内容
nickname	String	昵称
answerList	List	回复列表
isAnswer	boolean	是否为回复
isHasAnswer	boolean	是否有回复
time	String	时间戳
avatar	String	头像地址

调用示例：

```
mHtSdk.setHtDispatchQuestionListener(new HtDispatchQuestionListener() {
            @Override
            public void receiveQuestion(QuestionEntity questionEntity) {
                ...
            }
        });
```

投票

setHtVoteListener

设置投票监听

```
    void setHtVoteListener(HtVoteListener listener)
```

@参数说明：

参数	类型	描述
listener	HtVoteListener	投票监听回调

```
调用示例：
mHtSdk.setHtVoteListener(new HtVoteListener() {
            @Override
            public void voteStart(VoteEntity voteEntity) {
             //投票开始
            }

            @Override
            public void voteStop(VotePubEntity votePubEntity) {
            //投票结束
            }
        });
```

sendVote

发送投票

```
    void sendVote(String vid, String opts, Callback callback)
```

@参数说明：

参数	类型	描述
vid	String	投票id
opts	String	选择序号（从1开始算起）数组的字符串 ；例："[1,2]"
callback	Callback	回调

```
调用示例：
HtSdk.getInstance().sendVote(vid,"[1,2]", callback);
```

getAllVotes

获取全部投票数据

```
    void getAllVotes(int[] status, Callback callback)
```

@参数说明：

参数	类型	描述
status	int[]	获取数据状态数组，1:已发起投票状态（包括结束并未公布）2：投票结束并已公布
callback	Callback	回调

调用示例：

```
HtSdk.getInstance().getAllVotes(new int[]{1,2},new Callback(){
    void success(String result);//json格式的数据
    void failed(String failed);
});
```

getVotesUnreceived

获取未收到的的投票数据

```
    void getAllVotes(int[] status, Callback callback)
```

@参数说明：

参数	类型	描述
status	int[]	获取数据状态数组，1:已发起投票状态（包括结束并未公布）2：投票结束并已公布
callback	Callback	回调

调用示例：

```
HtSdk.getInstance().getVotesUnreceived(new Callback(){
    void success(String result);//json格式的数据
    void failed(String failed);
});
```

投票相关实体:

com.talkfun.sdk.module.VoteEntity

投票信息

@参数说明：

参数	类型	描述
vid	String	投票id
title	String	标题
imageUrl	String	标题
label	String	标签
nickname	String	发起者的昵称
startTime	String	发起投票时间
optional	int	投票的形式：单选 1,多选 2
opList	List	选项

com.talkfun.sdk.module.VoteOption

投票选项

@参数说明：

参数	类型	描述
content	String	内容

com.talkfun.sdk.module.VotePubEntity

投票结果

@参数说明：

参数	类型	描述
vid	String	投票id
title	String	标题
imageUrl	String	标题
label	String	标签
nickname	String	发起者的昵称
startTime	String	发起投票时间
endTime	String	结束投票时间
BriefVoteEntityList	List	各项投票结果
answer	String	投票答案，只有当选有答案时才有值，多项答案为"0,1,2"类似字符串，第一项索引为0
options	String	用户选择的选项，为"[1, 2]"类似字符串，第一项索引为1

com.talkfun.sdk.module.BriefVoteEntity

投票结果项

@参数说明：

参数	类型	描述
op	String	选项内容
opNum	int	选择该项人数
percent	int	选择该项比例

抽奖

抽奖监听：

com.talkfun.sdk.module.LotteryResult

抽奖结果

@参数说明：

参数	类型	描述
result	List	结果列表

com.talkfun.sdk.module.LotteryResult.ResultItem

抽奖结果项

@参数说明：

参数	类型	描述
xid	String	中奖人ID
launch_xid	String	发起人ID
nickname	NSString	中奖人名字
launch_nickname	NSString	发起人名字
roomid	String	房间id
liveid	NSString	直播ID
time	String	时间

```
调用示例：
HtSdk.getInstance().setHtLotteryListener(new HtLotteryListener() {
    @Override
    public void lotteryStart() {
        //开始抽奖
        ...
    }

    @Override
    public void lotteryStop(LotteryResult lotteryResult) {
        //结束抽奖
        ...
    }
});
```

鲜花

花朵事件监听:

```
调用示例：
HtSdk.getInstance().setHtDispatchFlowerListener(new HtDispatchFlowerListener() {
    @Override
    public void setFlowerNum(int num) {
    //花朵数量
        ...
    }
    @Override
    public void setFlowerLeftTime(int time) {
    //花朵获取剩余时间
        ...
    }
    @Override
    public void sendFlowerSuccess(String args) {
    //发送花朵成功
        ...
    }
});
```

送花:

```
调用示例：
HtSdk.getInstance().sendFlower();
```

公告

发布公告:

com.talkfun.sdk.module.NoticeEntity

公告

@参数说明：

参数	类型	描述
content	String	内容
time	String	公告发布时间

调用示例：

```
HtSdk.getInstance().setHtDispatchNoticeListener(new HtDispatchNoticeListener() {
    @Override
    public void receiveNotice(NoticeEntity noticeEntity) {
        ...
        }
    });
```

滚动通知:

com.talkfun.sdk.module.RollEntity

滚动通知

@参数说明：

参数	类型	描述
content	String	内容
time	String	时间
duration	int	时长
link	String	链接地址

调用示例：

```
HtSdk.getInstance().setHtDispatchRollAnnounceListener(new HtDispatchRollAnnounceListener(
) {
    @Override
    public void receiveRollAnnounce(RollEntity rollEntity) {
            ...
        }
    });
```

广播

广播:

com.talkfun.sdk.module.BroadcastEntity

@参数说明：

参数	类型	描述
uniqid	String	广播唯一标识
time	int	时间戳

调用示例：

```
HtSdk.getInstance().setHtBroadcastListener(new HtBroadcastListener() {
    @Override
    public void receiveBroadcast(BroadcastEntity broadcastEntity){
            ...
        }
    });
```

获取未收到的广播数据：

调用示例：

```
HtSdk.getInstance().getBroadcastsUnreceived(new Callback(){
    void success(String  result);//json 格式
    void failed(String failed);
});
```

获取全部的广播数据：

调用示例：

```
HtSdk.getInstance().getAllBroadcasts(new Callback(){
    void success(String  result);//json 格式
    void failed(String failed);
});
```

评分

发送评分：

@参数说明：

参数	类型	描述
contentScore	int	教学内容评分，区间[1,30]
methodScore	int	教学方法评分，区间[1,30]
effectScore	int	教学效果评分，区间[1,40]
msg	String	留言消息
callback	com.talkfun.sdk.event.Callback	回调

调用示例：

```
HtSdk.getInstance().sendScore(contentScore,methodScore,effectScore,msg,callback);
```

网络选择

- 如果当前网络速度不理想，SDK提供了接口更改网络
- 直播、伪直播和点播回放切换网络线路的接口统一化

```
//获取网络列表
- void getNetworkList(OnGetNetworkChoicesCallback listener)

//设置网络
- void setNetwork(int linePosition, NetItem item, OnSetNetworkCallback callback)
```

com.talkfun.sdk.module.NetWorkEntity

网络信息

@参数说明：

参数	类型	描述
cdnItems	String	CDN线路项信息列表
Network	NetworkInfo	网络信息，点播或伪直播该属性为空

com.talkfun.sdk.module.CDNItem

CDN线路项信息

@参数说明：

参数	类型	描述
operators	List	节点列表，点播或伪直播该属性为空
sourceName	String	来源

com.talkfun.sdk.module.NetItem

节点信息

- 点播或伪直播没有节点信息

@参数说明：

参数	类型	描述
name	String	名称
key	String	类型

com.talkfun.sdk.module.NetworkInfo

网络位置信息

- 点播或伪直播没有网络位置信息

@参数说明：

参数	类型	描述
location	String	位置
isp	String	运营商
ip	String	ip地址

调用示例：

```
//网路选择
//获取网络数据
HtSdk.getInstance().getNetworkList(new OnGetNetworkChoicesCallback() {
    @Override
    public void onGetChoicesSuccess(NetWorkEntity netWorkEntity){
        //获取成功
        ...
    }
    @Override
    public void onGetChoicesError(String msg) {
        //获取失败
        ...
    }
);

//设置网络
HtSdk.getInstance().setNetwork(linePosition, item, new OnSetNetworkCallback() {
    @Override
    public void onSwitchSuccess() {
        //设置成功
    }
    @Override
    public void onSwitchError() {
        //设置失败
    }
});
```

直播RTC接口说明

修订时间： 2020-05-13 10:23:22

Rtc

链接：

- 快速开始
- RTC模式初始化
- 画板
- 直播时长

rtc状态监听

调用示例：

```
mHtSdk.setRtcStatusListener(new OnRtcStatusListener() {
        /**
         * rtc开启
         */
        @Override
        public void onOpen() {
        }

        /**
         * rtc关闭
         */
        @Override
        public void onClose() {

        }

        /**
         * rtc中断
         */
        @Override
        public void onConnectionInterrupted() {

        }
        /**
         * rtc重连成功
         */
        @Override
        public void onReConnectSuccess() {
        }

    });
```

人员上麦相关

调用示例：

```
mHtSdk.setRtcMemberListener(new OnRtcMemberListener() {
    /**
     * 有用户被取消连麦的回调
     * @param rtcUserEntity 用户信息
     */
    @Override
    public void onKick(RtcUserEntity rtcUserEntity) {}

    /**
     * 有用户连麦（包含主播）的回调
     *
     * @param rtcUserEntity 用户信息
     * @param view 视频 view
     */
    @Override
   public void onUp(RtcUserEntity rtcUserEntity, View view) {}

    /**
     * 有用户取消连麦的回调
     * @param rtcUserEntity 用户信息
     */
    @Override
    public void onDown(RtcUserEntity rtcUserEntity) {}

    /**
     *
     * @param rtcUserEntity 用户信息
     * @param reason 离线原因
     */
    @Override
    public void onOffline(RtcUserEntity rtcUserEntity, int reason) {}

    /**
     * 学员被邀请连麦
     */
    @Override
    public void onInvite() {}

  /**
   * 取消邀请连麦
   */
    @Override
    public void onInviteCancel(){}

    /**
   * 拒绝申请连麦
   */
    @Override
    public  void rejectApply(){}
  /**
   * 更新视频
   * /
```

```
            void onUpdate(RtcUserEntity rtcUserEntity, View videoView);
        );
```

音视频状态监听

```
        mHtSdk.setRtcMediaStatusListener(new OnRtcMediaStatusListener() {
            @Override
         public void onVideoClose(RtcUserEntity rtcUserEntity) {
             //用户关闭视频
         }

             @Override
         public void onVideoOpen(RtcUserEntity rtcUserEntity) {
                //用户打开视频
         }

         @Override
         public void onAudioOpen(RtcUserEntity rtcUserEntity) {
             //用户打开音频
         }

         @Override
         public void onAudioClose(RtcUserEntity rtcUserEntity) {
             //用户关闭音频
         }

           /**
          * 全体音视频开关
          * @param status 具体描述查看com.talkfun.sdk.consts.GlobalStatus
          */
         @Override
         public void onGlobalStatus(int status){}
    });
```

Rtc操作类

（package com.talkfun.sdk.rtc.RtcOperatorProxy）

该类的获取时机应该在rtc开启后才调用，否则获取到的值为空

```
 RtcOperatorProxy  mRtcOperatorProxy =HtSdk.getInstance().getRtcOperatorProxy();

/**
 * 申请上讲台
 */
void apply(Callback<String> callback);
 /**
 * 主动下讲台
 */
void down(Callback<RtcUserEntity> callback);

/**
 * 取消上讲台申请
 */
void cancel(Callback<String> callback);


 /**
 * 关闭摄像头
 */
void closeVideo(Callback<RtcUserEntity> callback);

/**
 * 开启摄像头
 */
void openVideo(Callback<RtcUserEntity> callback);

/**
 * 麦克风 关闭静音
 */
void openAudio(Callback<RtcUserEntity> callback);

/**
 * 麦克风 打开静音
 */
void closeAudio(Callback<RtcUserEntity> callback);

/**
 *前后摄像头调换
 */
void swapVideo();

/**
 *邀请上讲台响应
 *
 *type 响应类型：accept：接受 ，reject:拒绝 推荐调用 Interaction.InviteStatus
 *callback 是否成功
 */
void respondInvite(String type, Callback callback)
```

Rtc错误监听

```
调用示例：
 mHtSdk.setRtcErrorListener(new OnRtcErrorListener() {
    @Override
    public void onError(int code, String msg) {

    }
});
```

Rtc 音量提示监听

```
调用示例：
mHtSdk.setRtcAudioVolumnListener(new OnRtcAudioVolumeListener() {
    @Override
    public void onVolumnIndication(List<AudioVolumeEntity> audioVolumeEntities) {

    }
});
```

获取Rtc初始化信息

```
RtcInfo mRtcInfo = mHtSdk.getRtcInfo();
```

---

com.talkfun.sdk.rtc.entity.RtcInfo

@参数说明：

参数	类型	描述
userApplyStatus	int	用户申请上讲台的状态
autoUp	int	1：自动上讲台 0：不自动上讲台
allowApply	boolean	是否允许申请上讲台

申请上麦状态

com.talkfun.sdk.rtc.consts.ApplyStatus

@参数说明：

参数	类型	描述
NO_APPLY	int	未申请
APPLYING	int	申请中
ALLOW	int	已允许

com.talkfun.sdk.rtc.entity.RtcUserEntity用户数据

@参数说明：

参数	类型	描述
xid	int	用户id
uid	String	合作方id
nickname	String	昵称
role	String	角色
time	int	时间
video	int	视频状态
audio	int	音频状态
score	int	奖励分数
drawPower	int	涂鸦权限

video 及audio状态

com.talkfun.sdk.rtc.consts.MediaStatus

参数	类型	描述
OPEN_FOR_ZHUBO	int	主播开启（1）
CLOSE_FOR_ZHUBO	int	主播关闭（0）
OPEN_FOR_USER	int	用户开启（11）
CLOSE_FOR_USER	int	用户关闭（10）

RtcUserEntity提供一些 公共方法

```
/**
 * 判断是否是当前用户
* @return boolean
 */
mRtcUserEntity.isMe()

/**
 * 判断视频是否开启
 * @return boolean
 */
mRtcUserEntity.isVideoOpen()

/**
 * 判断音频是否开启
 * @return boolean
 */
mRtcUserEntity.isAudioOpen()
```

AudioVolumeEntity

com.talkfun.sdk.rtc.entity.AudioVolumeEntity

参数	类型	描述
uid	int	用户id
volume	int	用户音量
totalVolume	int	总音量

全体音视频 控制状态值

com.talkfun.sdk.consts.GlobalStatus

参数	类型	描述
AUDIO_OPEN	int	全体音频开
AUDIO_CLOSE	int	全体音频关
VIDEO_OPEN	int	全体视频开
VIDEO_CLOSE	int	全体视频关
VIDEO_AUDIO_OPEN	int	全体音视频开
VIDEO_AUDIO_CLOSE	int	全体音视频关

画板

链接

- 快速开始
- RTC模式初始化
- RTC 相关接口调用
- 直播时长

画板操作类

```
IWhiteBoardOperator opetator = HtSdk.getInstance().getWhiteboardOperator();
```

调用示例：

```
/**
 * 设置画笔颜色
 *
 * @param color
 */
opetator.setPaintColor(int color);

/**
 * 设置画笔大小
 *
 * @param size
 */
opetator.setStrokeWidth(int size);

/**
 * 设置文字大小
 *
 * @return
 */
opetator.setTextSize(int size);

/**
 * 设置涂鸦类型
 *
 * @return
 */
opetator.setDrawType(int type);

/**
 * 撤退
 */
opetator.undoDrawable();

/**
 * 还原
 */
opetator.redoDrawable();
```

com.talkfun.sdk.whiteboard.config.DrawType 涂鸦类型

@参数说明：

参数	类型	描述
DRAW_PATH_MODE	int	曲线
DRAW_LINE_MODE	int	直线
DRAW_RECTANGLE_MODE	int	矩形
DRAW_OVAL_MODE	int	圆
DRAW_ARROW_MODE	int	箭头
DRAW_TEXT_MODE	int	文字
DRAW_DOTTED_LINE_MODE	int	虚线
DRAW_IMAGE_MODE	int	图片
DRAW_FIXED_MODE	int	混合
DRAW_CLEAR_MODE	int	清除

OnWhiteboardPowerListener

com.talkfun.sdk.rtc.interfaces.OnWhiteboardPowerListener

描述： 涂鸦权限监听类

成员方法说明

void onDrawEnable(int xid)

描述：赋予涂鸦功能 监听

参数说明

参数	描述
xid	获取涂鸦权限用户的id

void onDrawDisable(int xid)

描述：取消涂鸦功能监听

参数说明

参数	描述
xid	被取消涂鸦权限用户id

涂鸦权限参数说明

com.talkfun.sdk.rtc.consts.DrawPowerStatus

参数	类型	描述
OPEN	int	涂鸦打开（1）
CLOSE	int	涂鸦关闭（2）

示例

```
HtSdk.getInstance().setOnWhiteBoardLoadListener(new OnWhiteBoardLoadListener() {
            @Override
            public void onWhiteBoardLoad() {

            }
        });
```

OnWhiteBoardLoadListener

com.talkfun.sdk.presenter.OnWhiteBoardLoadListener

描述：白板ppt图片加载监听类

成员方法说明

void onWhiteBoardLoad()

描述：白板ppt图片加载成功回调方法

示例

```
HtSdk.getInstance().setOnWhiteBoardLoadListener(new OnWhiteBoardLoadListener() {
            @Override
            public void onWhiteBoardLoad() {

            }
        });
```

音视频

设置音视频监听

```
HtSdk.getInstance().setMultiMediaStatusChangeListener(
                        new OnMultiMediaStatusChangeListener() {
            @Override
            public void onMultiMediaApplicate(int id, int docType, String title, int
                        duration) {

            }

            @Override
            public void onMultiMediaStatusChange(int status, int time, String msg) {

            }
        })
```

OnMultiMediaStatusChangeListener

```
com.talkfun.sdk.event.OnMultiMediaStatusChangeListener
```

描述：音视频监听接口回调

成员方法

void onMultiMediaApplicate(int id, int docType, String title, int duration)

描述：音视频应用

参数说明

成员变量	参数类型	描述
id	int	音视频Id
docType	int	音视频类型(0:音频，1:视频)
tile	String	音视频标题
duration	int	音视频时长

com.talkfun.sdk.consts.DocType

参数描述

成员变量	参数类型	描述
MP3	int	音频（0）
MP4	int	视频（1）

onMultiMediaStatusChange(int status, int time, String msg)

描述：音视频状态监听

参数说明

成员变量	参数类型	描述
status	int	播放状态
time	int	当前播放时间点
msg	String	错误信息

status

参数说明

状态值	描述
STATUS_PLAY（1）	播放中
STATUS_PAUSE（2）	暂停
STATUS_SEEK（3）	跳转
STATUS_CLOSE（4）	音视频关闭
STATUS_COMPLATE（5）	音视频播放完成
STATUS_ERROR（6）	音视频播放错误

直播时长监听

链接

- 快速开始
- RTC模式初始化
- RTC相关接口调用
- 画板

```
mHtSdk.setLiveDurationListener(new OnLiveDurationListener() {
           @Override
            public void onTime(long totalTime) {}
} );
```

互动工具

1、 转盘监听

```
HtSdk.getInstance().setDialListener(new IInterAction.IDialListener() {
            @Override
            public void onStart(int angles, int res, int rotNum, boolean isEnd) {//开始旋转


            }

            @Override
            public void onApplicate() {//转盘应用

            }

            @Override
            public void onClose() {//转盘关闭

            }
        });
```

onStart(int angles, int res, int rotNum, boolean isEnd)

参数说明

参数	类型	描述
angles	int	转盘旋转角度
res	int	选中数字
rotNum	int	旋转次数
isEnd	boolean	是否已经旋转结束（针对于重新刷新或者进入刚进入直播）

2 、倒计时监听

```
HtSdk.getInstance().setTimerListener(new IInterAction.ITimerListener() {
            @Override
            public void onStart(int total, int duration) {//倒计时开始 total:总的倒计时长  单
                                                           //位s
                                                          //duration：剩余时长 单位s


            }

            @Override
            public void onPause(int duration) {//暂停  duration时长 单位s

            }

            @Override
            public void onEnd() {//倒计时结束

            }

            @Override
            public void onApplicate() {//倒计时应用

            }
              @Override
               public void onReReady() {//重新计时

            }
            @Override
            public void onClose() {//倒计时关闭

            }
        });
```

3、抢答器监听

```
HtSdk.getInstance().setResponderListener(new IInterAction.IResponderListener() {
            @Override
            public void onStart(long time) {//开始抢答，抢答时长

            }

            @Override
            public void onResponder(boolean isMe, String nickName, int duration) {
            //是否抢答成功
               //isMe:是否是当前用户
            //nickName:用户昵称
            //duration:时长 单位：s
            }

            @Override
            public void onEnd() {
            //抢答结束
            }

            @Override
            public void onApplicate() {
            //抢答器应用
            }

            @Override
            public void onClose() {
            //抢答器关闭
            }
        });
```

4、发起抢答

```
HtSdk.getInstance().sendResponder(duration, null);
//duration:抢答时长 单位：s
```

奖励

奖励监听

调用示例

```
HtSdk.getInstance().setAwardListener(OnAwardListener onAwardListener);
```

@参数说明

参数	类型	描述
onAwardListener	OnAwardListener	奖励回调

---

com.talkfun.sdk.rtc.interfaces.OnAwardListener

接口方法说明：

```
void receiveAward(AwardEntity awardEntity);
```

com.talkfun.sdk.rtc.entity.AwardEntity*

描述：奖励数据

@参数说明

参数	类型	描述
roomid	int	房间id
sendXid	int	发奖励用户的xid
toXid	int	接收到奖励的用户xid
goodsId	int	物品ID
amount	int	数量
toUid	String	被发送奖励的合作方id
toNick	String	被发送奖励的用户昵称
liveid	String	直播id
score	int	当前奖励新增分数
amountTotal	int	总数量
scoreTotal	int	总分数
goodsName	String	物品名称

直播接口说明

修订时间： 2021-05-26 17:43:29

点赞

点赞开关

- 在直播事件监听回调onLaunch()后获取LifeConfig配置，读取点赞开关配置LikeBean
- LikeBean.enable 值为1时开启，为0时关闭

```
调用示例：
LikeBean likeConfig = mSdk.getLifeConfig().global.switchX.like
boolean enable = likeConfig!= null ? likeConfig.enable == 1 : false;
```

sendLike

发送点赞

```
public void sendLike(int count,Callback<Integer> callback)
```

@参数说明：

参数	类型	描述
count	int	点赞数
callback	Callback	发送回调

```
调用示例：
mSdk.sendLike(1,new Callback<Integer>() {
        @Override
        public void success(Integer total)  {
        }

        @Override
        public void failed(String failed) {
        }
    });
```

setOnLikeListener

设置点赞监听

```
void setOnLikeListener(OnLikeListener listener)
```

@参数说明：

参数	类型	描述
listener	OnLikeListener	点赞监听

```
调用示例：
    mSdk.setOnLikeListener(new OnLikeListener() {
        @Override
        public void onReceiveLikeTotal(int total) {

        }
    });
```

商城

读取商城配置

- 在直播事件监听回调onLaunch()后获取LifeConfig配置，读取商城配置StoreBean

```
调用示例：
StoreBean storeConfig = mSdk.getLifeConfig().global.switchX.store
boolean enable = storeConfig != null ? storeConfig.enable == 1 : false;
```

商品列表

- 当商城配置StoreBean的类型type为1时，读取商品列表配置
- 读取LifeConfig配置的goods属性获取商品列表数据

```
调用示例：
List<GoodsBean> goodList = mSdk.getLifeConfig().goods
```

商城配置信息:

LifeConfig.GlobalBean.SwitchBean.StoreBean

商城配置

`@参数说明：`

参数	类型	描述
enable	int	是否开启 0未开启 1开启
type	int	类型;1商品列表,2商城链接,3商城二维码
data	DataBean	数据

LifeConfig.GlobalBean.SwitchBean.StoreBean.DataBean

商城配置数据

`@参数说明：`

参数	类型	描述
url	String	存储商城链接
qrcode	String	商城二维码链接

LifeConfig.GoodsBean 商品信息

`@参数说明：`

参数	类型	描述
id	int	商品id
name	String	商品名称
img	String	商品图片地址
price	String	商品现价
originalPrice	String	商品原价
tab	String	商品标签(特价:1, 限时:2, 新品:3, 钜惠:4, 秒杀:5)
putaway	int	商品状态：0下架，1上架，2推荐
pay	int	购买模式(1,链接购买 2,二维码购买)
qrcode	String	商品二维码链接
link_text	String	按钮链接文本

邀请榜单

邀请榜单开关

- 在直播事件监听回调onLaunch()后获取LifeConfig配置，读取邀请榜单开关配置InvitationListBean
- InvitationListBean.enable 值为1时开启，为0时关闭

```
    调用示例：
    InvitationListBean invitationListConfig = mSdk.getLifeConfig().global.switchX.invitat
ionList
    boolean enable = invitationListConfig != null ? invitationListConfig.enable == 1 : fa
lse;
```

setInvitationListPageSize

设置邀请榜单分页条数，默认10条

```
    void setInvitationListPageSize(int size)
```

@参数说明：

参数	类型	描述
size	int	分页条数

```
    调用示例：
    mSdk.setInvitationListPageSize(10);
```

getFirstPageInvitationList

获取邀请榜单首页列表

```
      void getFirstPageInvitationList(Callback<List<InvitationItem>> callback)
```

@参数说明：

参数	类型	描述
callback	Callback	回调

```
调用示例：
mSdk.getFirstPageInvitationList(new Callback<List<InvitationItem>>() {
        @Override
        public void success(List<InvitationItem> result) {
            subject.onSuccess(result);
        }

        @Override
        public void failed(String failed) {
            subject.onError(new Throwable(failed));
        }
    });
```

邀请信息：

InvitationItem

邀请信息

`@参数说明：`

参数	类型	描述
xid	int	用户唯一ID
nickname	String	用户昵称
avatar	String	用户头像地址
inviteCount	int	邀请数

举报

report

发送举报内容

```
      void  report(String content,Callback callback)
```

@参数说明：

参数	类型	描述
content	String	举报内容
callback	Callback	发送回调

```
    调用示例：
     mSdk.report(content, new Callback() {
            @Override
            public void success(Object result) {
            }

            @Override
            public void failed(String failed) {
            }
        });
```

点播接口说明

点播播放

初始化监听

- 通过HtSdk的setPlaybackListener()方法设置点播初始化监听
- 在SDK初始化完成后，调用HtSdk的onResume方法执行初始化数据加载
- 数据初始化执行完会调用PlaybackListener回调

```
调用示例：

HtSdk.getInstance().setPlaybackListener(new PlaybackListener() {
     @Override
     public void initSuccess() {
     //初始化完成
     }

     @Override
     public void onInitFail(String msg) {
     //初始化失败
     }

});
```

播放方法调用

播放

```
HtSdk.getInstance().playbackResume();
```

暂停

```
HtSdk.getInstance().playbackPause();
```

停止

```
HtSdk.getInstance().playbackStop();
```

跳转

@参数说明：

参数	类型	描述
time	long	跳转的时间（秒）

```
HtSdk.getInstance().playbackSeekTo(time);
```

重播

```
HtSdk.getInstance().replayVideo();
```

当前播放时间

```
HtSdk.getInstance().getVideoCurrentTime();
```

当前播放状态

```
HtSdk.getInstance().getVideoCurrentStatus();
```

播放专辑

参考专辑信息

点播信息

- 通过PlaybackInfo获取相关点播信息
- PlaybackInfo是一个单例
- 在初始化完成，调用PlaybackListener 的 initSuccess（）后获取

@参数说明：

参数	类型	描述
liveId	String	回放id
title	String	标题
durationLong	long	总时长
isAlbum	boolean	是否包含专辑
currentAlbumIndex	int	当前专辑索引

调用示例：

```
long duration = PlaybackInfo.getInstance().getDurationLong();
```

视频

配置摄像头视频

- 只需要HtSdk初始化时传入一个摄像头视频容器即可
- 课件模式播放摄像头视频时，SDK会生成摄像头视频播放器加到该容器

```
调用示例：

//1.首先获取容器对象和access_token值
//画板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 画板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
mHtSdk.init(pptContainer, videoViewContainer, access_token);

//4.（非必需）调用setWhiteboardViewContainer方法设置画板容器
FrameLayout pptContainer2;
//mHtSdk.setWhiteboardViewContainer(pptContainer2);
```

配置桌面分享/插播视频

- 初始化之后调用HtSdk的setDesktopVideoContainer方法设置桌面分享/插播视频容器
- 如果没调用setDesktopVideoContainer方法设置容器，默认使用画板容器，桌面分享的视频会添加到画板的上一层

```
调用示例：

HtSdk mHtSdk = HtSdk.getInstance();
mHtSdk.setDesktopVideoContainer(desktopVideoContainer);
```

视频监听

视频切换监听

- 当由课件模式或桌面分享/视频插播模式切换时，会调用该事件监听的onVideoModeChanging和onVideoModeChanged方法
- 当视频（摄像头视频和桌面分享/插播视频）开始、停止播放时调用onVideoStart和onVideoStop方法
- 当摄像头视频显示隐藏时调用onCameraShow和onCamerahide方法

com.talkfun.sdk.module.VideoModeType

视频模式类型

@类型常量说明：

类型常量	类型	描述
CAMERA_MODE	int	摄像头视频类型（静态变量）
DESKTOP_MODE	int	桌面分享视频类型（静态变量）
RTC_MODE	int	RTC 视频类型（静态变量）

```
调用示例：
HtSdk.getInstance().setOnVideoChangeListener(new OnVideoChangeListener() {
    @Override
    public void onVideoStart(int mode) {
    //视频开始播放
    }

    @Override
    public void onVideoStop(int mode) {
    //视频停止播放
    }

    @Override
    public void onVideoModeChanging(int beforeMode, int currentMode) {
    //视频模式切换中
    }

    @Override
    public void onVideoModeChanged() {
    //视频模式切换完成
    }

    @Override
    public void onCameraShow() {
    //摄像头显示
    }

    @Override
    public void onCameraHide() {
    //摄像头隐藏
    }
});
```

视频播放状态改变的监听回调

- 调用HtSdk的setOnVideoStatusChangeListener（）方法设置缓冲监听

com.talkfun.sdk.event.OnVideoStatusChangeListener 视频播放状态改变的监听回调

```
void onVideoStatusChange(int status, String msg)
设置可同时下载的线程数
参数：
    - status 状态
    - msg    信息（状态为STATUS_ERROR时有值，其他状态为空）
```

@状态常量说明：

状态常量	类型	描述
STATUS_PAUSE	int	暂停
STATUS_PLAYING	int	正在播放
STATUS_COMPLETED	int	播放完成
STATUS_ERROR	int	播放出错
STATUS_IDLE	int	停止播放
STATUS_BUFFERING	int	正在缓冲
STATUS_SEEKING	int	seek跳转中
STATUS_SEEKCOMPLETE	int	seek跳转完成

调用示例：

```
HtSdk.getInstance().setOnPlayerLoadStateChangeListener(new OnPlayerLoadStateChangeListene
r() {
    @Override
    public void onPlayerLoadStateChange(int loadState) {
    if (loadState == PlayerLoadState.MEDIA_INFO_BUFFERING_START) {
        Log.d(TAG, "缓冲开始");
    } else if (loadState == PlayerLoadState.MEDIA_INFO_BUFFERING_END) {
            Log.d(TAG, "缓冲结束");
        }
    }
});
```

视频缓冲监听

- 调用HtSdk的setOnPlayerLoadStateChangeListener（）方法设置缓冲监听

com.talkfun.sdk.event.OnPlayerLoadStateChangeListener 视频缓冲监听回调

调用示例：

```
HtSdk.getInstance().setOnPlayerLoadStateChangeListener(new OnPlayerLoadStateChangeListene
r() {
    @Override
    public void onPlayerLoadStateChange(int loadState) {
    if (loadState == PlayerLoadState.MEDIA_INFO_BUFFERING_START) {
        Log.d(TAG, "缓冲开始");
    } else if (loadState == PlayerLoadState.MEDIA_INFO_BUFFERING_END) {
            Log.d(TAG, "缓冲结束");
        }
    }
});
```

视频音量设置

初始化之后调用HtSdk的setPlayVolume(float volume)方法设置视频音量

调用示例:

```
 HtSdk.getInstance().setPlayVolume(1.0f);
```

画板

配置画板

- 只需要HtSdk初始化时传入一个画板容器即可
- 播放时SDK会生成一个画板视图加到该画板容器
- 也可在初始化之后调用HtSdk的setWhiteboardViewContainer方法设置切换画板容器

调用示例：

```
//1.首先获取容器对象和access_token值
//画板布局容器竖屏模式一般设置4:3
FrameLayout pptContainer;   // 画板布局容器
FrameLayout videoViewContainer;  // 摄像头视频播放器布局容器
String access_token; //直播access_token

//2.通过getInstance()方法获取HtSdk对象实例
HtSdk mHtSdk = HtSdk.getInstance();

//3.通过init()方法传1中的对象值初始化SDK
mHtSdk.init(pptContainer, videoViewContainer, access_token);

//4.（非必需）调用setWhiteboardViewContainer方法设置画板容器
FrameLayout pptContainer2;
//mHtSdk.setWhiteboardViewContainer(pptContainer2);
```

回放聊天

获取聊天数据

com.talkfun.sdk.module.ChatEntity

聊天信息

@参数说明：

参数	类型	描述
xid	String	用户唯一ID
uid	String	合作方用户ID
nickname	String	用户昵称
role	String	用户角色
gender	String	用户性别
avatar	String	用户头像地址
msg	String	消息内容
time	String	秒数

调用示例：

```
List<ChatEntity> list = PlaybackDataManage.getInstance().getChatList();
```

设置回放聊天获取数据监听

调用示例：

```
PlaybackDataManage.getInstance().setChatListener(new HtDispatchPlaybackMsgListener() {
    @Override
    public void getPlaybackMsgSuccess(int position) {
        //更新数据
        setChatList(PlaybackDataManage.getInstance().getChatList());
        if (position < chatMsgList.size()) {
            chatLv.setSelection(position);
        }
        else {
            chatLv.setSelection(chatMsgList.size() - 1);
        }
    }

    @Override
    public void getPlaybackMsgFail(String error) {
    //更新数据失败
    }
});
```

上拉加载更多聊天数据

调用示例：

```
PlaybackDataManage.getInstance().loadDownMoreData(PlaybackDataManage.DataType.CHAT);
```

下拉加载更多聊天数据

调用示例：

```
PlaybackDataManage.getInstance().loadDownMoreData(PlaybackDataManage.DataType.CHAT);
```

自动滚动跟随

调用示例：

```
AutoScrollListener autoScrollListener = new AutoScrollListener() {
    @Override
    public void scrollToItem(int pos) {
        if (isShow && chatAdapter != null) {
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {         setChatList(PlaybackDataManage.getInstance().getC
hatList());
                if (pos < chatMsgList.size()) {
                    chatLv.setSelection(pos);
                }
                else {
                chatLv.setSelection(chatMessageEntityList.size() - 1);
                }
            }
        });
        }
    }
};
PlaybackDataManage.getInstance().startAutoScroll(autoScrollListener, PlaybackDataManage.D
ataType.CHAT);
```

停止滚动跟随

调用示例：

```
PlaybackDataManage.getInstance().stopAutoScroll();
```

回放问答

获取问答数据

com.talkfun.sdk.module.QuestionEntity

问答信息

@参数说明：

参数	类型	描述
id	String	问答ID
xid	String	用户唯一ID
uid	String	合作方用户ID
replyId	String	回复的问题ID
role	String	角色
content	String	内容
nickname	String	昵称
answerList	List	回复列表
isAnswer	boolean	是否为回复
isHasAnswer	boolean	是否有回复
time	String	时间戳

调用示例：

```
List<QuestionEntity> list = PlaybackDataManage.getInstance().getRawQuestionList();
```

设置回放问答获取数据监听

调用示例：

```
PlaybackDataManage.getInstance().setQuestionListener(new HtDispatchPlaybackMsgListener()
{
    @Override
    public void getPlaybackMsgSuccess(int position) {
        //更新数据
        setQuestionList(PlaybackDataManage.getInstance().getRawQuestionList());
        if (position < questionMsgList.size()) {
            questionLv.setSelection(position);
        }
        else {
            questionLv.setSelection(questionMsgList.size() - 1);
        }
    }

    @Override
    public void getPlaybackMsgFail(String error) {
    //更新数据失败
    }
});
```

上拉加载更多问答数据

调用示例：

```
PlaybackDataManage.getInstance().loadDownMoreData(PlaybackDataManage.DataType.QUESTION);
```

下拉加载更多问答数据

调用示例：

```
PlaybackDataManage.getInstance().loadDownMoreData(PlaybackDataManage.DataType.QUESTION);
```

自动滚动跟随

调用示例：

```
AutoScrollListener autoScrollListener = new AutoScrollListener() {
    @Override
    public void scrollToItem(int pos) {
        if (isShow && chatAdapter != null) {
        getActivity().runOnUiThread(new Runnable() {
            @Override
            public void run() {         setQuestionList(PlaybackDataManage.getInstance().
getRawQuestionList());
                if (pos < questionMsgList.size()) {
                    questionLv.setSelection(pos);
                }
                else {
                    questionLv.setSelection(questionMsgList.size() - 1);
                }
            }
        });
        }
    }
};
PlaybackDataManage.getInstance().startAutoScroll(autoScrollListener, PlaybackDataManage.D
ataType.QUESTION);
```

停止滚动跟随

调用示例：

```
PlaybackDataManage.getInstance().stopAutoScroll();
```

专辑信息

获取专辑信息

- 调用PlaybackDataManage的getAlbumList()方法获取专辑信息列表

com.talkfun.sdk.module.AlbumItemEntity 专辑信息

@参数说明：

参数	类型	描述
id	String	回放id
accessToken	String	回放令牌
title	String	标题
thumbSrc	String	缩略图地址

调用示例：

```
List<AlbumItemEntity> albumItemEntities = PlaybackDataManage.getInstance().getAlbumList();
```

获取当前播放专辑索引

- 调用PlaybackInfo的getCurrentAlbumIndex()方法获取当前播放专辑索引

调用示例：

```
int currentIndex = PlaybackInfo.getInstance().getCurrentAlbumIndex();
```

播放专辑

- 从专辑信息列表获取相应的专辑信息
- 调用HtSdk的playAlbum()方法传入专辑信息播放该专辑

调用示例：

```
List<AlbumItemEntity> albumList = PlaybackDataManage.getInstance().getAlbumList();
int currentIndex = PlaybackInfo.getInstance().getCurrentAlbumIndex();
HtSdk.getInstance().playAlbum(albumList.get(currentIndex));
```

倍数播放

- SDK新增倍数播放功能，传入的数值为0 ~ 2之间的数值即可，传入0即为暂停。

```
//点播播放速度 (0-2]
void setPlaybackPlaySpeed(float speed)
```

片头广告

1. 添加广告容器

```
HtSdk.getInstance().setADVideoContainer(adVideoContainer);
```

参数说明

参数	类型	描述
adVideoContainer	ViewGroup	广告视频容器

2 设置广告容器配置

```
HtSdk.getInstance().setADVideoContainerConfig(videoViewConfig);
```

参数说明

参数	类型	描述
videoViewConfig	VideoViewConfig	视频容器配置

com.talkfun.sdk.config.VideoViewConfig 参数说明

参数	类型	描述
bgColor	int	视频容器背景颜色

3. 跳过广告片头

```
HtSdk.getInstance().skipAD();
```

4. 广告视频监听

```
HtSdk.getInstance().setADVideoListener(new OnADVideoListener() {
            @Override
            public void onADPrepare(ADConfig ADConfig) {//广告配置

            }

            @Override
            public void onADVideoStatusChange(int status, String msg) {//播放状态

            }

            @Override
            public void OnADCountDownTime(int time) {//倒计时时长(单位:秒)

            }
        });
```

com.talkfun.sdk.config.ADConfig

参数说明

参数	类型	描述
duration	int	广告总时长（单位：秒）
isSkipAD	boolean	是否跳过广告片头（true:是，false:否）

onADVideoStatusChange 说明

- status 状态分类

参数	类型	描述
STATUS_PAUSE（1）	int	播放暂停
STATUS_PLAYING（2）	int	播放中
STATUS_COMPLETED（3）	int	播放完成
STATUS_ERROR（4）	int	播放错误
STATUS_IDLE（5）	int	播放停止

- msg 描述：错误信息

回放广播

1.设置广播监听

```
PlaybackDataManage.getInstance().setBroadcastListener(new PlaybackBroadcastListener() {
            @Override
            public void onBroadcast(JSONObject jsonObject) {//与播放时间同步的广播。

            }
              @Override
             public void onBroadcastArray(JSONArray jsonArray){//当前课程所有的广播

            }
        });
```

注意： -注意：**JSONArray 是 org.json.JSONArray，而不是google的com.google.gson.JsonArray**

jsonObject 参数说明

参数	类型	描述
xid	String	用户唯一ID
uid	String	合作方用户ID
nickname	String	用户昵称
role	String	用户角色
time	String	时间(单位秒)
timestamp	String	时间戳
msg	String	内容
cmd	String	3：公共广播 4：自定义广播
gid	String	
status	String	状态
a	int	

离线播放

下载管理类

com.talkfun.sdk.offline.PlaybackDownloader

离线下载播放管理类，单例

```
public static PlaybackDownloader getInstance()
获取PlaybackDownloader实例

public void init(Context context)
初始化点播下载页

public void setDownLoadThreadSize(int mCorePoolSize)
设置可同时下载的线程数
参数：
     - mCorePoolSize 线程数

public void setRootFolder(String path)
设置下载文件存放目录
参数：
     - path 存放文件目录

public List<String> getPlaybackIdList()
获取下载队列的播放Id列表

public boolean containsID(String id)
点播是否已经在下载队列
参数：
     - id 点播id

public void appendDownloadTask(String token, String id, @Nullable String title,@Nullable
final String thumbnailImageUrl , PreDownLoad.OnappendDownloadListener listener)
添加下载任务
参数：
     - token 点播token
     - id 点播id
     - title 标题
     - thumbnailImageUrl 缩略图url
     - listener 添加下载任务事件回调

public ArrayList<DownloadInfoMode> getDownloadList()
获取下载任务信息列表

public DownloadInfoMode getDownloadInfo(String playbackID)
根据点播id获取对应的下载任务信息
参数：
 - id 点播id
获取点播任务信息

public Bitmap getThumbnailImage(String id, String url)
```

```
获取点播任务的缩略图
参数：
     - id 点播id
     - url 任务信息中缩略图url

public String getThumbnailPath(String id, String url)
获取离线缩略图的本地路径
参数：
     - id 点播id
     - url 缩略图url

public void startDownload(String playbackId)
开始下载
参数：
     - playbackId 下载点播id

public void pauseDownload(String playbackId);
暂停下载
参数：
     - playbackId 下载点播id

public void pauseAllDownload()
全部暂停下载

public void deleteDownload(String playbackId)
删除下载任务
参数：
     - playbackId 下载点播id

public void addDownLoadObserver(String playbackId,DownLoadManager.DownLoadObserver observ
er)
添加下载监听
参数：
     - observer 下载监听

public void removeObserver(String playbackId)
根据点播id移除下载监听

public void removeAllObserver()
移除所有下载监听

public void destroy()
```

com.talkfun.sdk.offline.http.DownLoadManager.DownLoadObserver 下载监听接口

```
void onDownLoadInfoChange(DownloadInfoMode info)
下载信息变化
参数:
     - info 下载任务信息列表
```

com.talkfun.sdk.offline.http.PreDownLoad.OnappendDownloadListener 添加下载任务事件回调接口

```
void success();
添加成功
void fail(int code, String msg);
添加失败
```

com.talkfun.sdk.offline.mode.DownloadInfoMode

下载任务信息

参数	类型	描述
id	String	回放id
title	String	标题
token	String	回放令牌
totalSize	long	总大小
finishSize	long	下载大小
totalNum	int	总文件数
finishNum	int	已下载文件数
duration	int	总时长，单位为秒（s）
thumbnailImageUrl	String	缩略图URL

com.talkfun.sdk.offline.PlaybackDownloader.Status 下载状态

```
STATE_UNDOWNLOAD 未下载
STATE_DOWNLOADING 正在下载
STATE_PAUSEDOWNLOAD 暂停下载
STATE_WAITINGDOWNLOAD 等待下载
STATE_DOWNLOADFAILED 下载失败
STATE_DOWNLOADED 下载完成
```

常见问题

自定义状态视图

- 初始化SDK时设置自定义状态视图

```
HtSdk mHtSdk = HtSdk.getInstance();

//设置直播未开始显示view
View liveWaitView =  layoutInflater.inflate(R.layout.live_wait_layout, null);
mHtSdk.setLiveWaitView(liveWaitView);

//设置直播结束显示view
View liveOverView = layoutInflater.inflate(R.layout.live_over_layout, null);
mHtSdk.setLiveOverView(liveOverView);

//设置正在加载初始化数据显示view
View loadingView = layoutInflater.inflate(R.layout.loading_layout, null);
mHtSdk.setLoadingView(loadingView);

//设置加载初始化数据失败显示view
View loadFailView = layoutInflater.inflate(R.layout.load_fail_layout, null);
mHtSdk.setLoadFailView(loadFailView);
```

混淆

- 为了确保SDK正常使用 ，请在 proguard-rules.pro 混淆文件中添加以下代码

```
-keep class com.talkfun.**{
    *;
}

-keep class fi.iki.elonen.**{
  *;
}

-keep class tv.danmaku.ijk.media.**{
  *;
}

#io.socket
-keep class io.socket.**{*;}
-keep interface io.socket.** { *; }
-keep class org.apache.commons.net.**{*;}


#retrofit2
-dontwarn retrofit2.**
-keep class retrofit2.** { *; }
-keep interface retrofit2.** { *; }
-keepattributes Signature
-keepattributes Exceptions

#okhttp3
-keepattributes Signature
-keepattributes Annotation
-keep class okhttp3.** { *; }
-keep interface okhttp3.** { *; }
-dontwarn okhttp3.**
-dontwarn okio.**

#RxJava RxAndroid
-dontwarn sun.misc.**
-keepclassmembers class rx.internal.util.unsafe.*ArrayQueue*Field* {
   long producerIndex;
   long consumerIndex;
}
-keepclassmembers class rx.internal.util.unsafe.BaseLinkedQueueProducerNodeRef {
    rx.internal.util.atomic.LinkedQueueNode producerNode;
}
-keepclassmembers class rx.internal.util.unsafe.BaseLinkedQueueConsumerNodeRef {
    rx.internal.util.atomic.LinkedQueueNode consumerNode;
}
```

```
#rtc
-keep class io.agora.**{*;}
-keep class com.tencent.**{*;}

#xlog
-keep class com.tencent.mars.** { *; }
-keep class razerdp.basepopup.** { *; }
```

如何在按Home键退到后台后保持继续播放

```
//进入后台是否暂停（默认是暂停）
HtSdk.getInstance().setPauseInBackground(false);
```

java.lang.NoClassDefFoundError错误

如果在运行app时有类似如下的异常信息

```
java.lang.NoClassDefFoundError: Failed resolution of: Lcom/talkfun/sdk/http/a;
at com.talkfun.sdk.log.TalkFunLogger.init(Unknown Source)
at com.talkfun.sdk.a.a(Unknown Source)
at com.talkfun.sdk.c.a(Unknown Source)
at com.talkfun.sdk.c.<init>(Unknown Source)
at com.talkfun.sdk.HtSdk.init(Unknown Source)
at com.talkfun.cloudlive.activity.LiveNativeActivity.initView(LiveNativeActivity.java:177
)
at com.talkfun.cloudlive.activity.BasePlayActivity.onCreate(BasePlayActivity.java:92)
at com.talkfun.cloudlive.activity.LiveNativeActivity.onCreate(LiveNativeActivity.java:153)
```

原因是sdk中的相应代码调用了第三方库的类，而项目中没有引入该库，运行时找不到某个类报的异常。

解决方法：

检查是否有添加sdk所依赖的第三方库

关于ReactNative、APICloud或其他非原生语言的接入

- 该SDK使用原生语言编写
- 如果使用的是H5项目，请使用H5相关SDK
- 如果使用非官方原生语言构建APP项目，由于未提供其他非原生SDK，如ReactNative、APICloud等，需要自定义一个中间件，使SDK提供的接口可供项目调用。具体请参考相关语言如何调用原生接口代码

SDK下载

- SDK版本记录
- Version 3.8.0（2022-4-22)

  1、生活直播模板增加宣传图历史查看入口

  2、直播新增答题卡功能

  SDK下载
- Version 3.7.8（2021-11-11)

  1、支持展示直播宣传图功能

  2、聊天、公告内容支持超链接识别跳转
- Version 3.7.7（2021-8-27)

  1、直播直播支持聊天@回复、删除指定聊天内容

  2、大班回放点播记录播放时间点优化

  3、修复一些已知问题
- Version 3.7.6（2021-7-29)

  1、小班直播根据后台配置项进行设置

  2、小班直播一些交互逻辑优化和样式调整

  3、生活直播投票逻辑优化

  4、修复一些已知问题
- Version 3.7.5（2021-6-22)

  1、生活直播支持视频插播、桌面分享

  2、支持播放暖场视频

  3、优化虚拟机器人聊天信息显示头像

  4、修复一些已知问题
- Version 3.7.4（2021-5-27)

  1、优化小班样式

  2、优化生活直播模板

  3、修复一些已知问题
- Version 3.7.3（2021-4-27)

  1、支持主讲模式、画廊模式等视频布局

2、设备状态优化

- Version 3.7.1（2021-1-22)

1、支持低延迟大班直播

- Version 3.7.0（2021-1-11)

1、直播视频播放优化

- Version 3.6.2（2020-12-7)

1、画板支持显示动态PPT

2、更新RTC库版本

3、离线下载状态优化

- Version 3.6.1（2020-12-1)

1、新增支持直播模式切换视频流不断流

- Version 3.6.0（2020-10-27)

1、签到优化

2、白板优化

- Version 3.5.9（2020-8-21)

1、聊天数据支持解析自定义扩展字段

2、更新升级RTC版本

3、修复一些问题

- Version 3.5.8（2020-8-4)

1、1v1逻辑优化

2、视频加载超时处理

- Version 3.5.7（2020-7-24)

1、支持人数分组显示

2、优化网络稳定性检测

3、优化UI层

- Version 3.5.6（2020-7-16)

1、大班互动和小班文案调整

2、添加监听主播拒绝用户连麦申请

3、抽奖效果优化

4、读取配置是否开启举手功能

- Version 3.5.5（2020-7-6)

1、处理大班互动与小班桌面分享切换显示问题

2、修复一些Bug

- Version 3.5.4（2020-6-28)

  1、优化大班互动上下讲台切换逻辑

  2、调整大班抽奖UI

  3、修复一些Bug

- Version 3.5.3（2020-5-28)

  1、大班互动和小班支持聊天接收图片

  2、大班互动和小班新增网络状态显示

  3、小班上讲台用户支持批量音频视频开关操作

  4、小班支持显示公告/广播/通知

  5、更新支持AndroidX

- Version 3.5.2（2020-5-14)

  1、大班互动与小班直播页面UI改版

  2、大班互动与小班直播新增线路选择功能

  3、大班互动与小班直播新增提问功能

  4、修复一些Bug

- Version 3.5.1（2020-4-15)

  1、新增大班互动交互功能

  2、优化UI逻辑

  3、升级RTC库

- Version 3.5.0（2020-3-25)

  1、优化小班统计

  2、优化小班逻辑

- Version 3.4.7（2020-3-9)

  1、大班互动与小班直播课支持前后摄像头切换

  2、云课堂小班支持弹幕

- Version 3.4.6（2020-2-20)

  1、大班互动与小班直播课支持点名签到

  2、UI层支持大班互动与小班视频画板全屏

  3、大班互动与小班直播课支持互动工具交互

4、修复一些Bug

- Version 3.4.5（2020-1-3)

  1、大班互动直播的音视频与涂鸦同步优化

  2、UI层支持禁言用户删除该用户之前聊天记录

  3、其他UI层优化

- Version 3.4.4（2019-10-30)

  1、直播的音视频与涂鸦同步优化

  2、点播添加seek状态回调及获取当前播放状态

  3、视频重连优化

- Version 3.4.3（2019-9-11)

  1、视频播放优化

  2、修复一些bug

- Version 3.4.2（2019-8-28)

  1、支持回放播放片头视频

  2、支持回放广播

- Version 3.4.1（2019-8-15)

  1、支持播放音视频素材

  2、支持被邀请上讲台

- Version 3.4.0（2019-7-9)

  1、更新1V16页面模板

  2、小班新增投票、抽奖功能

  3、文字交互配置

  4、动态评分配置

  5、新增白板页区域回调接口

  6、UI层优化

- Version 3.3.4（2019-6-6)

  1、新增1v1直播模式

  2、修复一些bug

- Version 3.3.3（2019-5-20)

  1、修复伪直播视频加载问题

  2、聊天分组过滤

3、添加响应课前课后是否可发送聊天的配置

4、修复图片素材层级错位问题

- Version 3.3.2（2019-3-20)

  1、支持自动上讲台

  2、优化直播逻辑

  3、优化UI层代码

- Version 3.3.1（2019-1-29)

  1、班课互动和视频连麦模式功能优化

  2、处理一些bug

- Version 3.3.0

  1、 直播新增支持视频连麦模式

- Version 3.2.0

  1、 直播新增支持班课互动模式

- Version 3.1.5（2019-6-19)

  1、优化socket连接重连机制

  2、修复一些bug

  SDK下载

- Version 3.1.4（2019-5-20)

  1、修复伪直播视频加载问题

  2、聊天分组过滤

  3、添加响应课前课后是否可发送聊天的配置

  4、修复图片素材层级错位问题

- Version 3.1.3（2019-2-18)

  1、资源超时重试加载优化

  2、视频重试机制优化

  3、日志统计优化

- Version 3.1.2（2019-1-29)

  1、SDK指令解析优化

  2、直播支持审核通过后提问的显示、添加删除提问的功能

  3、统计优化

4、直播观看人数增加机器人数

5、增加防盗录水印显示

6、修复UI层的一些BUG

- Version 3.1.1（2018-8-27)

1、点播视频播放优化

2、增加直播评分接口

3、增加进入直播显示最近部分聊天信息支持

4、修复一些bug

- Version 3.1.0（2018-7-20)

1、日志优化

2、网络请求优化

3、增加课程模式获取直播起始时间

4、修复一些bug

- Version 3.0.9（2018-6-14)

1、修复课程模式课程结束状态

```
2、修复word文档滚动涂鸦位置偏移

3、优化离线下载

4、点播回放资源释放优化
```

- Version 3.0.8（2018-5-29)

  1、修复一些bug

- Version 3.0.7（2018-5-15)

  1、聊天信息优化

  2、PPT图片加载优化

- Version 3.0.6（2018-4-23)

  1、增加日志检查

- Version 3.0.5（2018-2-2)

  1、修复权限切换后涂鸦未擦除的bug

- Version 3.0.4（2018-1-25)

  1、sdk优化

  2、修复一些bug

- Version 3.0.3（2018-1-16)

  1、伪直播支持

- Version 3.0.2（2018-1-05)

  1、更新升级第三方依赖库

  2、修复一些bug

- Version 3.0.1（2017-12-29)

  1、修复点播擦除涂鸦bug

- Version 3.0.0（2017-12-25)

  1、sdk重构

2、画板渲染性能提升

3、重构UI层代码

4、修复已知bug

- Version 1.8.0

1、增加投票广播获取接口

2、支持HTTPS

3、增加设置离线信息数据库保存路径

4、重构UI层代码

5、修复一些bug

- Version 1.7.9

1、离线下载优化

2、修复一些bug

- Version 1.7.8

1、增加支持后台播放设置

2、点播新增播放停止和立即跳转方法

3、离线下载优化

- Version 1.7.7

1、直播点播新增视频加载状态回调

2、点播问答优化

- Version 1.7.6.3

  1、点播倍速播放优化

- Version 1.7.6.2

  1、聊天提问增加头像显示

2、投票答案显示

3、修复Bug

- Version 1.7.6.1

  1、增加鉴权接口

- Version 1.7.6

  1、直播增加全体禁言广播

2、直播增加图片投票支持

3、修复Bug

- Version 1.7.5

  1、新增网络选择切换接口

2、离线下载播放一些优化

3、点播视频提供倍速播放

- Version 1.7.4.7

  1、修复sdk bug

2、更新demo