目錄

简介

欢拓云直播iOS SDK 是一套基于 armv7、armv7s、arm64 处理器设备的应用程序接口，发起直播功能，开放式UI自定义，能够快速让您的APP拥有直播以及互动功能。

功能列表

* 创建课程：根据讲师的需要设定课程主题、课程开始时间以及结束时间
* 视频直播：可以切换手机前后置摄像头进行直播
* PPT展示：可以将云端的素材，导入到PPT展示区域，发起直播后，可以向用户展示PPT
* PPT涂鸦：支持再ppt区域划动进行涂鸦，轻轻松松做板书
* 图片导入：选择手机照片导入到ppt区域
* 语音模式：关闭摄像头进行纯音频直播
* 静音模式：关闭直播声音
* 聊天功能：在聊天区域用户进行文字聊天
* 问答功能：在问答区域回复用户的问题

兼容性

支持iOS 8.0及以上操作系统，支持armv7、armv7s、arm64处理器。

账号获取

* 账号:管理后台获取的 5 位纯数字的主播 ID
* 密码:管理后台设置的主播密码

注意事项

您在使用中遇到任何问题，都可以通过邮件反馈给我们。邮件地址：luoliuyou@talk-fun.com

修订时间： 2020-05-27 14:16:39

快速开始

模块说明

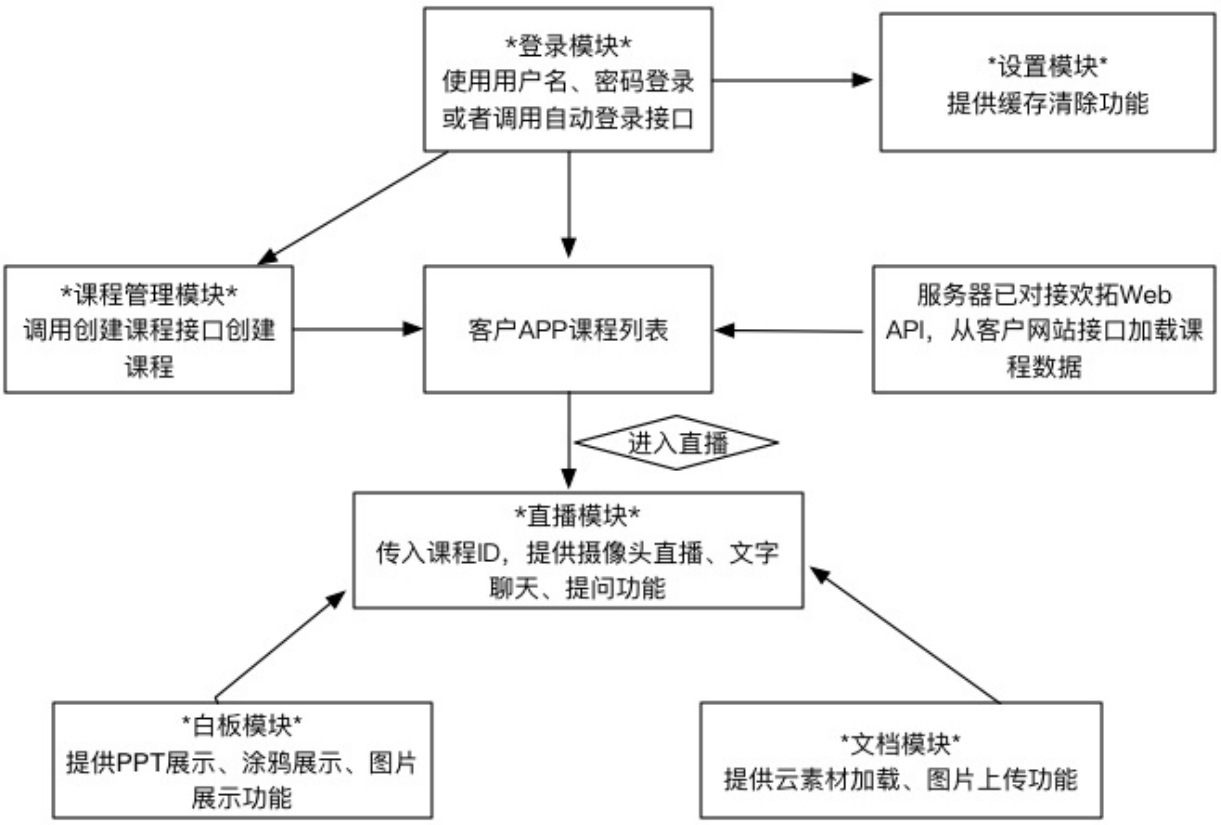
欢拓云直播SDK提供了以下模块：

课程回放上传模块：上传直播的回放
文档上传模块：可以上传设备的文档
登录模块：该模块提供用户登录及鉴权接口，提供输入账号密码登录及使用token自动登录。
课程模块：该模块提供课程管理功能，包括创建课程，查看课程。
直播模块：该模块提供直播功能，如开启摄像头，开启PPT，与用户互动聊天、回复用户提问等。
白板模块：该模块提供白板展示PPT、涂鸦功能。
文档模块：该模块提供加载云端素材，上传用户图片到云端的功能。
设置模块：该模块提供用户设置功能。
工具模块：工具类

业务流程说明

- 调用登录模块，完成用户登录
- 调用课程模块，创建课程
- 调用直播模块
    - 直播相关的操作，如开启摄像头直播，与用户聊天互动
    - 调用文档模块，可以载入PPT
    - 调用白板模块，可以展示PPT、涂鸦

流程图：



配置开发环境

配置开发环境 1. 使用**cocoapods**或者手动引入以下第三方依赖

- platform :ios, '9.0'
- target '你的项目名字' do
  - use_frameworks! #欢拓直播SDK的依赖
  - pod 'CloudLiveSDKFramework', '~> 2.9.9' #欢拓直播SDK

end

"TalkfunLive.h"（直播间相关方法）

1.初始化大班sdk

```
    self.liveManager = [TalkfunLive shareInstance];
    self.liveManager.delegate = self;

    TalkfunLiveInitParams *params = [[TalkfunLiveInitParams alloc]init];
    params.courseID = self.model.course_id;
    [self.liveManager enterRoom:params];
```

2.开始直播

```
  [self.liveManager startLive:^(NSDictionary* data) {

            if ([data[@"code"] intValue] == TalkfunCloudLiveCodeSuccess) {



            }
            else
            {


                    if ([data[@"code"] intValue] == TalkfunCloudLiveCodeInTheLive) {
                        [weakSelf.view alert:@"不能进行直播" message:data[@"msg"]];
                    }
                    else{
                        [weakSelf createDismissTimerWithTitle:data[@"msg"]];
                    }


            }

        }];

        其它事件使用同理   ....................
```

监听事件示例 **(on:callback:)**

```
- (void)on:(NSString *)event callback:(void (^)(id result))callback;
```

监听初始化房间信息事件示例 **(on:callback:)**

```
 [self.liveManager on:TALKFUN_EVENT_ROOM_INIT callback:^(id result) {
        NSLog(@"%@:%@",TALKFUN_EVENT_ROOM_INIT,result);
    }];
```

聊天信息示例 **(emit:params:callback:)**

```
        [self.liveManager emit:TALKFUN_EVENT_CHAT_SENT params:parameter callback:^(id res
ult) {
            if ([result[@"code"] intValue] == TalkfunCloudLiveCodeSuccess) {
                [[NSNotificationCenter defaultCenter] postNotificationName:TALKFUN_NOTIFI
CATION_CHAT_SEND object:nil userInfo:result[@"data"]];
            }
        }];
```

监听违规直播时的回调示例 **(on:callback:)**

```
    [self.liveManager on:TALKFUN_EVENT_ZHUBO_KICKED callback:^(id result) {
        //结束直播
        [weakSelf.liveManager  stop:^(id result) {

            //提示
            dispatch_async(dispatch_get_main_queue(), ^{
                [weakSelf presentViewController:weakSelf.caveat animated:YES completion:n
il];
            });

        }];

    }];



 其它事件监听使用同理    ...................
```

```
```

所有在线人员(包含主播)

字段	类型	必须	说明
page	int	N	页码(默认:1)
size	int	N	条数(默认:100，最大值100)

```
调用示例:
   TFMemberParams *params = [[TFMemberParams alloc]init];
    params.page = 1;
    params.size = 100;

 [self.rtcEngineKit getMemberList:params callback:^(NSInteger code, NSMutableArray<TFMemb
er *> * _Nullable list, TFError * _Nullable error) {

        }];
```

所有学员 (不包含主播)

字段	类型	必须	说明
page	int	N	页码(默认:1)
size	int	N	条数(默认:100，最大值100)

```
调用示例:
   TFMemberParams *params = [[TFMemberParams alloc]init];
    params.page = 1;
    params.size = 100;

 [self.rtcEngineKit getUserList:params callback:^(NSInteger code, NSMutableArray<TFMember
 *> * _Nullable list, TFError * _Nullable error) {

        }];
```

当前用户进入直播间 (代理方法)

```
- (void)onMemberJoinMe:(TFMember*_Nullable)result
{

}
```

其他人进入直播间 (代理方法)

```
- (void)onMemberJoinOther:(TFMember*_Nullable)result
{

}
```

有人退出房间 (代理方法)

```
- (void)onMemberLeave:(TFMember*_Nullable)result
{

}
```

同时在线人数 (代理方法)

```
- (void)onPeakConcurrentUsers:(TFMember*_Nullable)result
{

}
```

----------下面的代码为过期的方法,不建议使用----------

所有在线人员数据

字段	类型	必须	说明
page	int	N	页码(默认:1)
size	int	N	条数(默认:100，最大值100)

返回值类型为NSArray:

参数	类型	描述
xid	NSNumber	欢拓用户ID
uid	NSString	合作方用户ID
nickname	NSString	用户昵称
role	NSString	用户角色
gender	NSNumber	用户性别
avatar	NSString	用户头像地址
chat	NSDictionary	聊天属性
voice	NSDictionary	声音属性

调用示例:

```
[self.liveManager emit:TALKFUN_EVENT_MEMBER_LIST params:@{@"page":@(1),@"size":@(100)} ca
llback:(^ id result){

}];
```

当前用户进入直播间 **TALKFUN_EVENT_MEMBER_JOIN_ME**

total参数:在线总人数 (NSNumber) member参数:

参数	类型	描述
xid	NSNumber	欢拓用户ID
uid	NSString	合作方用户ID
nickname	NSString	用户昵称
role	NSString	用户角色
gender	NSNumber	用户性别
avatar	NSString	用户头像地址
chat	NSDictionary	聊天属性
voice	NSDictionary	声音属性

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_MEMBER_JOIN_ME callback:^(id result) {


}];
```

- 其他人进入直播间 member:join:other (TALKFUN_EVENT_MEMBER_JOIN_OTHER)

total参数:在线总人数（NSNumber) member参数:

参数	类型	描述
xid	NSNumber	欢拓用户ID
uid	NSString	合作方用户ID
nickname	NSString	用户昵称
role	NSString	用户角色
gender	NSNumber	用户性别
avatar	NSString	用户头像地址
chat	NSDictionary	聊天属性
voice	NSDictionary	声音属性

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_MEMBER_JOIN_OTHER callback:^(id result) {


}];
```

- 直播间在线人数 member:total (TALKFUN_EVENT_MEMBER_TOTAL)

total参数:在线总人数（NSNumber) member参数:

参数	类型	描述
total	NSNumber	总人数

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_MEMBER_TOTAL callback:^(id result) {


}];
```

- 其他人退出房间 TALKFUN_EVENT_MEMBER_LEAVE

参数:在线总人数（NSNumber)

参数	类型	描述
nickname	NSString	用户名称
total	NSNumber	现有总人数
uid	NSString	用户UID
xid	NSNumber	用户欢拓唯一ID

```
调用示例：
[self.liveManager on: TALKFUN_EVENT_MEMBER_LEAVE callback:^(id result) {


}];
```

发送聊天

```
调用示例：
 [self.liveManager sendChat:@"聊天内容"callback:^(NSInteger code, TFChatData * _Nullable m
odel, TFError * _Nullable error) {
            if (code ==0 ) {

            }
    }];
```

接收别人的聊天事件回调 (代理方法)

```
调用示例：
- (void)onDidReceiveText:(TFChatData*_Nullable)result
{

}
```

----------下面的代码为过期的方法,不建议使用----------

发送事件示例 **(emit:params:callback:)**

- 公共聊天 chat:send (TALKFUN_EVENT_CHAT_SENT)

创建一个信息字典 parameter: @{@"msg":@"我是信息内容"}

```
调用示例：
[self.liveManager emit:TALKFUN_EVENT_CHAT_SENT params:parameter callback:^(id result) {
            if ([result[@"code"] intValue] == 0) {


            }else{
                //发送失败，提示  result[@"msg"]
            }
}];
```

- 私聊 chat:private (TALKFUN_EVENT_CHAT_PRIVATE)

创建一个信息字典 parameter: @{@"xid":123,@"msg":@"我是信息内容"}

```
调用示例：
[self.liveManager emit:TALKFUN_EVENT_CHAT_PRIVATE params:parameter callback:^(id result)
{
            if ([result[@"code"] intValue] == 0) {


            }else{
                //发送失败，提示  result[@"msg"]
            }
}];
```

- 禁言 chat:disable (TALKFUN_EVENT_CHAT_DISABLE)

`@参数说明:`

参数	类型	描述
xid	NSNumber	用户ID
nickname	NSString	用户名字

```
调用示例：
[self.talkfunSDK on:@" chat:disable " withCallback:^(id obj) {
//返回数据
{
"xid":337861,
"nickname":"rrrr"
}
}];```


##### 监听事件示例 (on:callback:)

* 文字信息 chat:send (TALKFUN_EVENT_CHAT_SENT)

| 参数          | 类型        | 描述                    |
| -------    | ------- | ----------------    |
|xid         |NSNumber    |用户唯一ID                |
|uid        |NSString    |合作方用户ID            |
|nickname    |NSString    |用户昵称                |
|role        |NSString    |用户角色                |
|gender    |NSNumber    |用户性别                |
|avatar    |NSString    |用户头像地址            |
|msg        |NSString    |消息内容                |
| chat        | NSDictionary    | 聊天属性        |
|time        |NSNumber    |时间戳(从1970-01-01到现在的秒数)|

调用示例： [self.liveManager on:TALKFUN_EVENT_CHAT_SENT callback:^(id result) {

}];
```

```
* 文字信息 chat:private (TALKFUN_EVENT_CHAT_PRIVATE)

| 参数          | 类型        | 描述                    |
| -------    | ------- | ----------------    |
|xid         |NSNumber    |用户唯一ID               |
|uid        |NSString    |合作方用户ID            |
|nickname    |NSString    |用户昵称                |
|role        |NSString    |用户角色                |
|gender    |NSNumber    |用户性别                |
|avatar    |NSString    |用户头像地址            |
|msg        |NSString    |消息内容                |
| chat        | NSDictionary    | 聊天属性        |
|time        |NSNumber    |时间戳(从1970-01-01到现在的秒数)|
```

调用示例： [self.liveManager on:TALKFUN_EVENT_CHAT_PRIVATE callback:^(id result) {

}]; ```

回复别人的提问

```
    TalkfunQuestionReplyParameter *model = [[TalkfunQuestionReplyParameter alloc]init];
        model.course_id = self.model.course_id;
        model.replyId = self.replyModel.qid;
        model.content = self.replyTFView.myTextField.text;
        WeakSelf
        [self.liveManager questionReply:model callback:^(NSInteger code, TFQuestionData *
_Nullable model, TFError * _Nullable error) {

            if (code ==0 ) {

            }

        }];
```

接收别人的提问 (代理方法)

```
- (void)onQuestionAsk:(TFQuestionData*_Nonnull)result
{

}
```

助教回复了提问 (代理方法)

```
- (void)onQuestionReply:(TFQuestionData*_Nonnull)result
{

}
```

----------下面的代码为过期的方法,不建议使用----------

主播问题回复学员 (emit:params:callback:)

- question:reply (TALKFUN_EVENT_QUESTION_REPLY) 创建一个信息字典 parameter: @{@"content":@"我是信息内容",@"replyId":@"我是replyId",@"course_id":@"我是course_id"}

```
调用示例：
[self.liveManager emit:TALKFUN_EVENT_QUESTION_REPLY params:parameter callback:^(id result
) {


}];
```

接收提问 (on:callback:) TALKFUN_EVENT_QUESTION_ASK

参数	类型	描述
content	NSString	提问内容
xid	NSNumber	用户唯一ID
avatar	NSString	用户头像地址
uid	NSString	合作方用户ID
qid	NSString	提问ID
nickname	NSString	用户昵称
role	NSString	角色
sn	NSNumber	提问顺序
time	NSNumber	时间戳(从1970-01-01到当前的秒数)

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_QUESTION_ASK callback:^(id result) {


}];
```

助教问题回复学员**(on:callback:) TALKFUN_EVENT_QUESTION_REPLY**

参数	类型	描述
content	NSString	回复的内容
replyId	NSString	被回复的提问的ID
qid	NSString	回复的ID
nickname	NSString	回复者昵称
xid	NSNumber	欢拓用户ID
avatar	NSString	用户头像地址
uid	NSString	合作方用户ID
role	NSString	用户角色
time	NSNumber	时间戳
question	NSDictionary	回答的问题

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_QUESTION_REPLY callback:^(id result) {


}];
```

接收别人的送花 (代理方法)

```
调用示例：
- (void)onFlowerSend:(TFChatData*_Nullable)result
{

}
```

----------下面的代码为过期的方法,不建议使用----------

鲜花

- 送花 flower:send (TALKFUN_EVENT_FLOWER_SEND)

参数	类型	描述
amount	NSNumber	花朵总数
nickname	NSString	用户名
role	NSString	用户角色
sendtime	NSNumber	送花时间
time	NSString	现在的时间
uid	NSString	合作方用户ID
xid	NSNumber	用户ID
avatar	NSString	用户头像地址

```
调用示例：
[self.liveManager on:TALKFUN_EVENT_FLOWER_SEND callback:^(id result) {

        [[NSNotificationCenter defaultCenter] postNotificationName:TALKFUN_NOTIFICATION_F
LOWER_SEND object:nil userInfo:result];

}];
```

"TalkfunLive.h"（直播间相关方法）

1.初始化小班sdk

```
    //初始化RTC
    self.rtcEngineKit = [TalkfunRtcEngineKit shareInstance];

    TFRtcEngineKitInitParams * parms = [[TFRtcEngineKitInitParams alloc]init];
    parms.courseID = self.model.course_id;
    self.rtcEngineKit.delegate = self;

    [self.rtcEngineKit enterRoom:parms];
```

监听事件示例 (on:callback:)

```
- (void)on:(NSString *)event callback:(void (^)(id result))callback;
```

监听初始化房间信息事件示例 (on:callback:)

```
   [self.rtcEngineKit on:TALKFUN_EVENT_ROOM_INIT callback:^(id result) {


    }];
```

聊天信息示例 (emit:params:callback:)

```
        [self.rtcEngineKit emit:TALKFUN_EVENT_CHAT_SENT params:parameter callback:^(id re
sult) {

            if ([result[@"code"] intValue] == TalkfunCloudLiveCodeSuccess) {


            }

        }];
```

接收上讲台的用户视图view (on:callback:)

```
 //TODO:允许上讲台
    [self.rtcEngineKit on:TALKFUN_EVENT_RTC_USER_UP callback:^(id obj) {
//         NSLog(@"允许上讲台的用户数据%@",obj);

        PERFORM_IN_MAIN_QUEUE(

                              UIView       *view  = obj[@"view"];
                              NSDictionary *dict  = obj[@"userData"];


                                [weakSelf.videoArea addVideoContainer:view userData:dict];

                                [weakSelf.UserList UpdateData:dict tpye:TalkfunUserListMode
lStatePlatform];


                                [weakSelf drawCount];

                                )

    }];


 其它事件监听使用同理    ...................
```

```
```

讲台上的用户列表

```
调用示例：
NSArray * upUserArr= [self.rtcEngineKit optimized_getRtcUserEntityList];
```

申请上讲台的用户列表

```
调用示例：
NSArray * upUserArr= [self.rtcEngineKit optimized_getRtcUserEntityList];
```

获取在线的用户列表

```
调用示例：

   TFMemberParams *params = [[TFMemberParams alloc]init];
   params.page = 1;
   params.size = 100;

 [self.liveManager getMemberList:params callback:^(NSInteger code, NSMutableArray<TFMembe
r *> * _Nullable list, TFError * _Nullable error) {



    }];
```

获取邀请上讲台的列表

```
调用示例：
    [self.rtcEngineKit optimized_getInviteList:^(NSInteger code, NSMutableArray<TFMember
*> * _Nullable list, TFError * _Nullable error) {

    }];
```

用户申请上讲台(代理方法)

```
调用示例：
-(void)rtcApply:(TFMember*_Nullable)result
{

}
```

用户取消申请上讲台(代理方法)

```
调用示例：
-(void)rtcCancelApply:(TFMember*_Nullable)result
{

}
```

允许上讲台(代理方法)

```
调用示例：
- (void)rtcUp:(TalkfunRtcModel*)model
{

}
```

主动下讲台(代理方法)

```
调用示例：
- (void)rtcDown:(TalkfunRtcModel*)model
{

}
```

被踢下讲台(代理方法)

```
调用示例：
- (void)rtcKick:(TalkfunRtcModel*)model
{

}
```

用户离线(代理方法)

```
调用示例：
- (void)rtcOffline:(TalkfunRtcModel*)model
{

}
```

打开摄像头(代理方法)

```
调用示例：
- (void)rtcOpenVideo:(TalkfunRtcModel*)model
{

}
```

关闭摄像头(代理方法)

```
调用示例：
- (void)rtcCloseVideo:(TalkfunRtcModel*)model
{

}
```

打开麦克风(代理方法)

```
调用示例：
- (void)rtcOpenAudio:(TalkfunRtcModel*)model
{

}
```

关闭麦克风(代理方法)

```
调用示例：
- (void)rtcCloseAudio:(TalkfunRtcModel*)model
{

}
```

打开涂鸦权限(代理方法)

```
调用示例：
- (void)rtcDrawEnable:(TalkfunRtcModel*)model
{

}
```

关闭涂鸦权限(代理方法)

```
调用示例：
- (void)rtcDrawDisable:(TalkfunRtcModel*)model
{

}
```

让用户下讲台

```
调用示例：
   [self.rtcEngineKit kick:xid callback:^(NSInteger code, TFError * _Nullable error) {

    }];
```

- 公告

参数	类型	描述
message	NSString	发送文字内容

```
调用示例：
[self.rtcEngineKit sendAnnouncement:message callback:^(NSInteger code, TFError * _Nullabl
e error) {

}];
```

- 通知
- model 参数

参数	类型	描述
link	NSString	超链接，需要HTTP开头
duration	NSInteger	选择需要的时长，
content	NSString	输入的通知内容

```
调用示例：
[self.rtcEngineKit sendNotice:model callback:^(NSInteger code, TFError * _Nullable error)
 {

}];
```

- 广播

参数	类型	描述
message	NSString	发送文字内容
tag	NSInteger	是否自动发送，默认为1

```
调用示例：
    TFBroadCastParams * parms = [[TFBroadCastParams alloc]init];
    parms.msg = message;
    parms.__auto = @(tag);

    [self.rtcEngineKit sendBroadCast:parms callback:^(NSInteger code, TFBroadcast * _Null
able result, TFError * _Nullable error) {



       }];
```

- 点名列表

```
调用示例：
    [self.rtcEngineKit getRollRecordList:^(NSInteger code, NSMutableArray<TFRtcEngineKitR
ollList *> * _Nullable list, TFError * _Nullable error) {



    }];
```

- 发起点名

参数	类型	描述
startTime	NSInteger	发起点名的时间时长

```
调用示例：
    [self.rtcEngineKit startRoll:selectTime callback:^(NSInteger code, TFError * _Nullabl
e error) {


    }];
```

- 查看点名详情
- model 参数

参数	类型	描述
signId	NSString	点名ID
page	NSInteger	点名页数
signUser	NSInteger	是否只获取已签到的用户0或不传，获取全部用户；1 只获取已签到用户
total	NSInteger	总数

```
调用示例：
    [self.rtcEngineKit checkRollDetail:model callback:^(NSInteger code, TFRtcEngineKitSig
nNews * _Nullable result, TFError * _Nullable error) {




    }];
```

- 结束点名

参数	类型	描述
signID	NSString	点名ID

```
调用示例：
    [self.rtcEngineKit endRoll:signID callback:^(NSInteger code, TFError * _Nullable erro
r) {



    }];
```

- 抽奖记录

```
调用示例：
    [self.rtcEngineKit getLotteryList:^(NSInteger code, NSMutableArray<TFRtcEngineKitLott
ery *> * _Nullable list, TFError * _Nullable error) {


    }];
```

- 抽奖开始

```
调用示例：
    [self.rtcEngineKit startLottery:^(NSInteger code, TFError * _Nullable error) {




    }];
```

- 结束抽奖

参数	类型	描述
count	NSString	抽奖人数

```
调用示例：
    [self.rtcEngineKit stopLottery:count callback:^(NSInteger code, TFError * _Nullable e
rror) {



      }];
```

- 投票列表

```
调用示例：
    [self.rtcEngineKit getVoteList:^(NSInteger code, NSMutableArray<TFRtcEngineKitVote *>
 * _Nullable list, TFError * _Nullable error) {


    }];
```

- 删除投票

参数	类型	描述
vid	NSString	投票ID

```
调用示例：
  [self.rtcEngineKit deleteVote:vid callback:^(NSInteger code, TFError * _Nullable error)
 {




  }];
```

- 查看投票详情

参数	类型	描述
vid	NSString	投票ID

```
调用示例：
    [self.rtcEngineKit checkVoteDetail:vid callback:^(NSInteger code, TFRtcEngineKitVoteE
mit * _Nullable result, TFError * _Nullable error) {



       }];
```

- 结束投票

参数	类型	描述
vid	NSString	投票ID
ispublic	BOOL	是否公开，默认为YES

调用示例：

```
   [self.rtcEngineKit endVote:vid andisPublic:ispublic callback:^(NSInteger code, TFError
 * _Nullable error) {



            }];
```

- 预投票，已保存，再发起投票

参数	类型	描述
vid	NSString	投票ID

调用示例：

```
    [self.rtcEngineKit publishSavedVote:vid callback:^(NSInteger code, TFError * _Nullabl
e error) {



    }];
```

- 保存投票
- model

参数	类型	描述
title	NSString	投票的标题
label	NSString	投票的标签
type	NSString	投票的类型
optional	NSString	投票的可选项数量
op	NSDictionary	投票的全部选项
imageUrl	NSString	题目图片地址，没有则传空字符串
answer	NSString	问题答案，多个答案用英文逗号分隔
vid	NSString	要修改的投票ID

调用示例：

```
        [self.rtcEngineKit saveVote:model callback:^(NSInteger code, NSString * _Nullable
 vid, TFError * _Nullable error) {



         }];
```

- 添加投票
- model

参数	类型	描述
title	NSString	投票的标题
label	NSString	投票的标签
type	NSString	投票的类型
optional	NSString	投票的可选项数量
op	NSDictionary	投票的全部选项
imageUrl	NSString	题目图片地址，没有则传空字符串
answer	NSString	问题答案，多个答案用英文逗号分隔
vid	NSString	要修改的投票ID

调用示例：

```
  [self.rtcEngineKit publishVote:model callback:^(NSInteger code, NSString * _Nullable vi
d, TFError * _Nullable error) {



           }];
```

- 抢答器 互动工具类型。

参数	类型	描述
status	NSInteger	枚举

调用示例：

```
    TFInteractiveToolResponderParams *params = [[TFInteractiveToolResponderParams alloc]i
nit];
    params.status =  operateType;
    [self.rtcEngineKit responderConfiguration:params callback:^(NSInteger code, TFError *
 _Nullable error) {



    }];
```

- 定时器 互动工具类型

参数	类型	描述
lengthTime	NSInteger	总时长
remainingTime	NSInteger	剩余的时长
status	NSInteger	状态

```
调用示例：
    TFInteractiveToolTimeParams *params = [[TFInteractiveToolTimeParams alloc]init];
    params.lengthTime = length;
    params.remainingTime = time;
    params.status =  operateType;

    [self.rtcEngineKit timerConfiguration:params callback:^(NSInteger code, TFError * _Nu
llable error) {



     }];
```

- 转盘 互动工具类型

```
转盘应用：
   [self.rtcEngineKit turntableApplicate:^(NSInteger code, TFError * _Nullable error) {

    }];
转盘开始：
    TFTurntableParams *params = [[TFTurntableParams alloc]init];
    params.degrees = degrees ;
    params.answer = answer ;
    params.count = count;
    [self.rtcEngineKit turntableStart:params callback:^(NSInteger code, TFError * _Nullab
le error) {

    }];
转盘关闭：
[self.rtcEngineKit turntableStop:^(NSInteger code, TFError * _Nullable error) {

}];
```

- 前后摄像头切换

```
[self.rtcEngineKit switchCamera];
```

- 打开摄像头

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
    [self.rtcEngineKit openVideo:xid callback:^(NSInteger code, TFError * _Nullable error
) {


    }];
```

- 关闭摄像头

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
    [self.rtcEngineKit closeVideo:xid callback:^(NSInteger code, TFError * _Nullable erro
r) {


    }];
```

- 打开麦克风

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
    [self.rtcEngineKit openAudio:xid callback:^(NSInteger code, TFError * _Nullable error
) {


    }];
```

- 关闭麦克风

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
    [self.rtcEngineKit closeAudio:xid callback:^(NSInteger code, TFError * _Nullable erro
r) {


    }];
```

- 关闭所有摄像头

```
调用示例：
   [self.rtcEngineKit closeAllCamera:^(NSInteger code, TFError * _Nullable error) {



    }];
```

- 开启所有摄像头

```
调用示例：
   [self.rtcEngineKit openAllCamera:^(NSInteger code, TFError * _Nullable error)   {



    }];
```

- 关闭所有麦克风

```
调用示例：
   [self.rtcEngineKit closeAudio:^(NSInteger code, TFError * _Nullable error) {



    }];
```

- 开启所有麦克风

```
调用示例：
   [self.rtcEngineKit openAllAudio:^(NSInteger code, TFError * _Nullable error) {



    }];
```

- 邀请连麦

参数	类型	描述
xid	NSString	用户xid

```
调用示例：
 [self.rtcEngineKit giveInvite:button.xid callback:^(NSInteger code, TFError * _Nullable
error) {


   }];
```

- 取消邀请连麦

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
 [self.rtcEngineKit cancelInvite:button.xid callback:^(NSInteger code, TFError * _Nullabl
e error) {


   }];
```

- 拒绝用户申请连麦

参数	类型	描述
xid	NSString	用户xid

调用示例：

```
 [self.rtcEngineKit rejectApply:button.xid callback:^(NSInteger code, TFError * _Nullable
 error) {

   }];
```

- 奖励

参数	类型	描述
score	NSString	每次获取的分数，默认为1
xid	NSString	用户xid
goodsId	NSString	商品xid

调用示例：

```
  TFInteractiveToolAwardParams *params = [[TFInteractiveToolAwardParams alloc]init];
  params.score = @"1";
  params.xid = [weakSelf getNumber:button.xid];
  params.goodsId = @"1";
   [self.rtcEngineKit award:params callback:^(NSInteger code, TFInteractiveToolAwardData
* _Nonnull data, TFError * _Nullable error) {



           }];
```

- 获取学员列表用户奖励分数

```
调用示例：
    [self.rtcEngineKit getAwardList:^(NSInteger code, NSMutableArray<TFInteractiveToolAwa
rdData *> * _Nullable list, TFError * _Nullable error) {

      }];
```

- 设置美颜 YES：有美颜效果；NO：无美颜

参数	类型	描述
enable	BOOL	YES ：有美颜，NO:无美颜

```
调用示例：
[self.rtcEngineKit setBeautyEffectOptions: enable];
```

- 静音/取消所有用户的声音

参数	类型	描述
enable	BOOL	YES ：静音，NO:取消静音

```
调用示例：
 [self.rtcEngineKit muteAllRemoteAudio: enable];
```

- 镜像 ,NO为镜像,YES不镜像

参数	类型	描述
mirror	BOOL	YES ：不镜像，NO:镜像

```
调用示例：
[self.rtcEngineKit setLocalViewMirror: mirror];
```

----------下面的代码为过期的方法,不建议使用----------

- 讲台上的用户列表

```
 NSArray *UpUsers =  [self.RtcEngineKit getRtcUserEntityList];
```

- 获取申请上讲台列表

```
    NSArray*ApplyListData = [self.RtcEngineKit getRtcApplyList];
    [self.UserList initUserData:ApplyListData] ;
```

- 获取在线列表

```
    [self.RtcEngineKit getMemberList:^(id result) {

    }];
```

- 有用户申请上讲台

```
    [self.RtcEngineKit on:TALKFUN_EVENT_RTC_APPLY callback:^(id obj) {



    }];
```

参数	类型	描述
xid	NSNumber	用户ID

- 允许用户上讲台

```
      [self.rtcEngineKit up:xid callback:^(NSInteger code, TFError * _Nullable error) {

       }];
```

@参数说明:

参数	类型	描述
xid	NSNumber	用户ID

多媒体操作

- 打开麦克风

```
[self.rtcEngineKit openAudio:xid callback:^(NSInteger code, TFError * _Nullable error) {

            }];
```

- 关闭麦克风

```
     [self.rtcEngineKit closeAudio:xid callback:^(NSInteger code, TFError * _Nullable err
or) {


            }];
```

- 打开摄像头

```
 [self.rtcEngineKit openVideo:btn.xid callback:^(NSInteger code, TFError * _Nullable erro
r) {


            }];
```

- 关闭摄像头

```
           [self.rtcEngineKit closeVideo:btn.xid callback:^(NSInteger code, TFError * _Nu
llable error) {


            }];
```

常见问题

如何完成鉴权

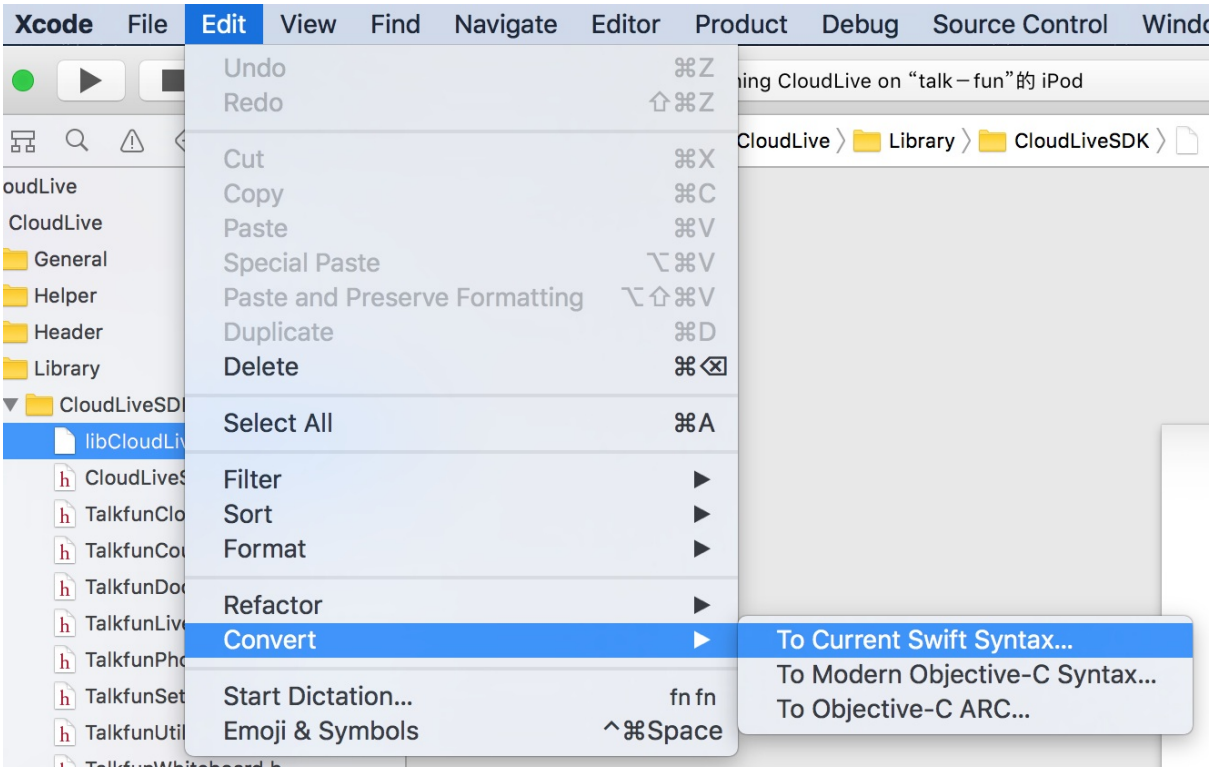
- 需要使用主播的账号密码或者登录token，完成鉴权才可以操作其它模块。
- 流程如下：

```
    1、调用TalkfunCloudLive模块，调用isLogin方法判断是否已经登录，如果未登录或者需要使用其它账号进行登录，继续下面步骤

    2、客户对接服务器端接口，获取主播账号或者登录token

    3、调用TalkfunCloudLive模块，调用login或者autoLogin方法完成登录鉴权
```

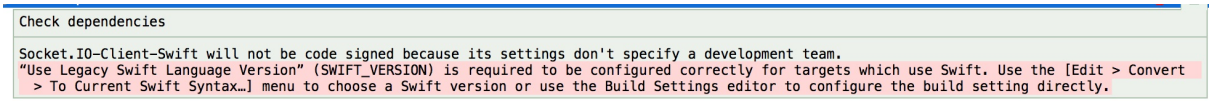
- 参考文档： 自动登录

框架集成问题

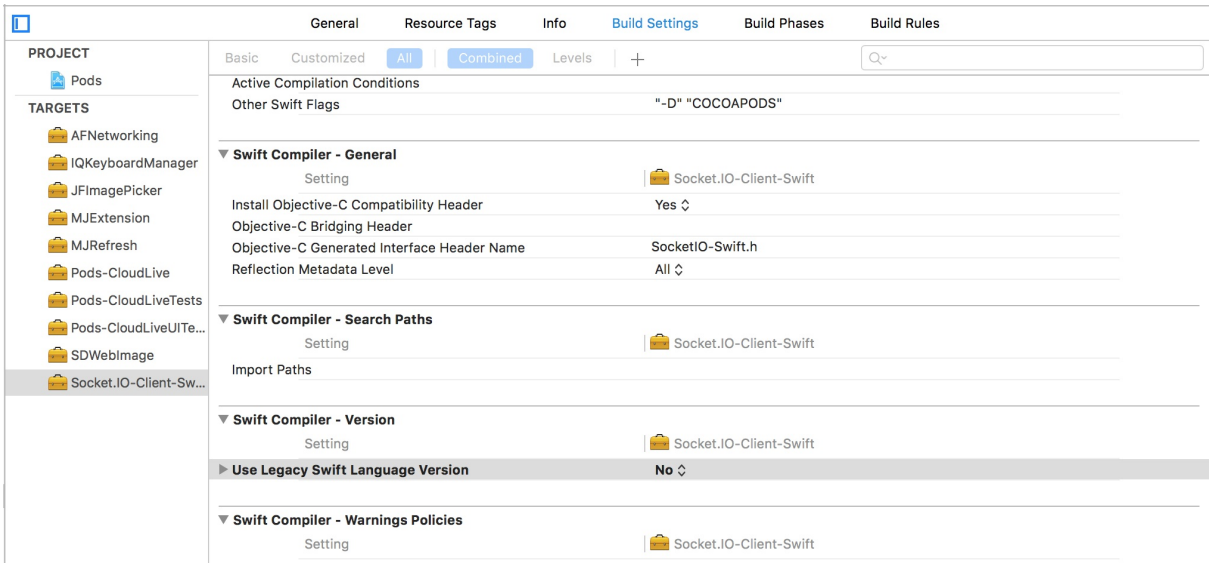
- Socket.IO-Client-Swift 自动或手动转换到最新的swift语法(convert to current swift syntax)



- 编译时 Socket.IO-Client-Swift 出错



解决方法:



- Socket.IO-Client-Swift 手动引入

  ### Manually (iOS 7+)

  1. Copy the Source folder into your Xcode project. (Make sure you add the files to your target(s))
  2. If you plan on using this from Objective-C, read this on exposing Swift code to Objective-C.

常见错误

- Terminating app due to uncaught exception 'NSInvalidArgumentException', reason: '-[__NSArrayM lfPopFirstObject]: unrecognized selector sent to instance 0x170441f80'

  ```
  解决方法：
  Other Linker Flags  添加 -all_load
  ```

- Terminating app due to uncaught exception 'NSInternalInconsistencyException', reason: '* -[AVAssetWriter startWriting] Cannot call method when status is 3'

  ```
  解决方法：
  设备已经占用了，有可能是在退出直播控制器的时候，没有调用 [liveManager stopLive] 方法
  ```

网络状态处理

直播过程中网络状态变化如何处理

目前SDK内部并未进行网络状态的监听，主要是网络状态改变后的相关操作，需要由用户自己决定。
因此为了弱化改部分与SDK的耦合，请在APP自行进行网络状态的监听

```
// 网络
typedef enum: NSUInteger{
    TalkfunCloudLiveNetworkStatusNone = 0,          //没有网络
    TalkfunCloudLiveNetworkStatusWifi,          //wifi
    TalkfunCloudLiveNetworkStatusWWAN,          //WWAN
    TalkfunCloudLiveNetworkStatusOther        //其它
}TalkfunCloudLiveNetworkStatus;

//网络状态改变，将networkStatus 属性设置为对应状态即可
self.liveManager.networkStatus = TalkfunCloudLiveNetworkStatusWifi;
```

云直播

```
调用的时候只需要引入CloudLiveSDK.h文件
#import "CloudLiveSDK.h"
```

SDK版本号

```
extern NSString * const TalkfunCloudLiveSDKVersion;     //SDK版本
```

状态码说明

```
typedef NS_ENUM (NSInteger,TalkfunCloudLiveCode){
TalkfunCloudLiveCodeGetDataFail = -101,
TalkfunCloudLiveCodeWrongPassword = -3,
TalkfunCloudLiveCodeFail = -1,
TalkfunCloudLiveCodeSuccess = 0,
TalkfunCloudLiveCodeInTheProcessing = 34,
TalkfunCloudLiveCodeInTheLive  = 204,
TalkfunCloudLiveCodeNoExist   = 300,
TalkfunCloudLiveCodeUploading = 500,
TalkfunCloudLiveCodeHasBeenUploaded = 501,
TalkfunCloudLiveCodeUploadFail = 503,
TalkfunCloudLiveCodeUnknownError = 1000
};
```

聊天状态

```
typedef NS_ENUM(NSInteger,TalkfunCloudLiveChatStatus){
TalkfunCloudLiveChatStateDisable = 0,       //禁言状态
TalkfunCloudLiveChatStateEnable = 1         //允许聊天状态
};
```

全体禁言状态

```
// 全体禁言状态
typedef NS_ENUM(NSInteger,TalkfunCloudLiveChatDisableAllStatus){
TalkfunCloudLiveChatDisableAllStatusClose = 0,      //全体禁言关闭
TalkfunCloudLiveChatDisableAllStatusOpen = 1        //全体禁言开启
};
```

网络

```
// 网络
typedef enum: NSUInteger{
TalkfunCloudLiveNetworkStatusNone = 0,          //没有网络
TalkfunCloudLiveNetworkStatusWifi,          //wifi
TalkfunCloudLiveNetworkStatusWWAN,          //WWAN
TalkfunCloudLiveNetworkStatusOther        //其它
}TalkfunCloudLiveNetworkStatus;
```

用户角色说明

```
extern NSString * const TalkfunCloudLiveMemberRoleSpadmin;       //超级管理员(老师)
extern NSString * const TalkfunCloudLiveMemberRoleAdmin;         //管理员(助教)
extern NSString * const TalkfunCloudLiveMemberRoleUser;          //普通用户(学生)
extern NSString * const TalkfunCloudLiveMemberRoleGuest;         //游客
```

系统事件说明

```
extern NSString * const TALKFUN_EVENT_CONNECT;              //连接
extern NSString * const TALKFUN_EVENT_RECONNECT;            //重新连接
extern NSString * const TALKFUN_EVENT_RECONNECT_ATTEMPT;    //尝试重新连接
extern NSString * const TALKFUN_EVENT_DISCONNECT;           //断开
extern NSString * const TALKFUN_EVENT_ERROR;                //错误
```

用户事件说明

```
extern NSString * const TALKFUN_EVENT_ROOM_INIT;            //初始化房间信息

extern NSString * const TALKFUN_EVENT_MEMBER_TOTAL;         //总人数
extern NSString * const TALKFUN_EVENT_MEMBER_JOIN_ME;       //自己进入房间
extern NSString * const TALKFUN_EVENT_MEMBER_JOIN_OTHER;    //其他人进入房间
extern NSString * const TALKFUN_EVENT_MEMBER_LEAVE;         //其他人退出房间
extern NSString * const TALKFUN_EVENT_MEMBER_LIST;          //用户列表

extern NSString * const TALKFUN_EVENT_CHAT_SENT;            //聊天信息
extern NSString * const TALKFUN_EVENT_CHAT_PRIVATE;         //私聊
extern NSString * const TALKFUN_EVENT_CHAT_DISABLE;         //禁言
extern NSString * const TALKFUN_EVENT_CHAT_DISABLE_ALL;     //全部禁言

extern NSString * const TALKFUN_EVENT_QUESTION_ASK;         //提问信息
extern NSString * const TALKFUN_EVENT_QUESTION_REPLY;       //回复信息
extern NSString * const TALKFUN_EVENT_QUESTION_DELETE;      //提问删除

extern NSString * const TALKFUN_EVENT_FLOWER_SEND;          //送花信息

extern NSString * const TALKFUN_EVENT_VOTE_NEW;             //发起投票
extern NSString * const TALKFUN_EVENT_VOTE_PUB;             //结束投票

extern NSString * const TALKFUN_EVENT_LOTTERY_START;        //开始抽奖
extern NSString * const TALKFUN_EVENT_LOTTERY_STOP;         //停止抽奖

extern NSString * const TALKFUN_EVENT_ANNOUNCE_NOTICE;      //公告
extern NSString * const TALKFUN_EVENT_ANNOUNCE_ROLL;        //滚动通知

extern NSString * const TALKFUN_EVENT_BROADCAST;            //广播信息
```

语音相关消息

```
extern NSString * const TALKFUN_EVENT_VOICE_MODE_CHANGE;    //语音模式切换
extern NSString * const TALKFUN_EVENT_VOICE_POWER_ALLOW;    //允许发言
extern NSString * const TALKFUN_EVENT_VOICE_POWER_FORBID;   //禁止发言

extern NSString * const TALKFUN_EVENT_VOICE_QUEUE_CONTROL;  //麦序队列状体控制
extern NSString * const TALKFUN_EVENT_VOICE_QUEUE_TIME;     //麦序时间设置
extern NSString * const TALKFUN_EVENT_VOICE_QUEUE_MOVE;     //移动位置(废弃)
extern NSString * const TALKFUN_EVENT_VOICE_QUEUE_MOVETO;   //移动麦序到第二位置
extern NSString * const TALKFUN_EVENT_VOICE_QUEUE_CLEAR;    //清空麦序

extern NSString * const TALKFUN_EVENT_VOICE_HAND_UP;        //举手
extern NSString * const TALKFUN_EVENT_VOICE_HAND_ALLOW;     //举手允许发言
extern NSString * const TALKFUN_EVENT_VOICE_HAND_FORBID;    //举手关闭发言
extern NSString * const TALKFUN_EVENT_VOICE_HAND_REMOVE;    //从举手列表移除
extern NSString * const TALKFUN_EVENT_VOICE_HAND_CLEAR;     //清空举手列表
extern NSString * const TALKFUN_EVENT_VOICE_HAND_LEAVE;     //用户退出举手列表
```

广播事件说明

```
extern NSString * const TALKFUN_NOTIFICATION_TOTAL_ONLINE;      //在线人数
extern NSString * const TALKFUN_NOTIFICATION_CHAT_SEND;         //聊天信息
extern NSString * const TALKFUN_NOTIFICATION_QUESTION_ASK;      //提问信息
extern NSString * const TALKFUN_NOTIFICATION_QUESTION_REPLY;    //回复信息
extern NSString * const TALKFUN_NOTIFICATION_FLOWER_SEND;       //送花信息
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_DOWNLOAD_START;       //文件下载开始
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_DOWNLOAD_PROGRESS;    //文件下载进度
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_DOWNLOAD_DONE;        //文件下载完毕
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_DOWNLOAD_FAIL;        //文件下载失败

extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_START;         //文件上传开始
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_PROGRESS;      //文件上传进度
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_DONE;          //文件上传成功
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_FAIL;          //文件上传失败

extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_PROCESS_START;        //文件处理开始
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_PROCESS_PROGRESS;     //文件处理进度
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_PROCESS_DONE;         //文件处理成功
extern NSString * const TALKFUN_NOTIFICATION_DOCUMENT_PROCESS_FAIL;         //文件处理失败

extern NSString * const TALKFUN_NOTIFICATION_WHITEBOARD_RELOAD;             //白板重新加载
extern NSString * const TALKFUN_NOTIFICATION_WHITEBOARD_COMMAND_SEND;       //白板命令发送
```

头文件说明

"CloudLiveSDK.h"

```
引入SDK相关头文件、定义有关事件名称
请求返回的数据@"code"的格式说明：0(TalkfunCloudLiveCodeSuccess)为成功，其他值为失败
```

"TalkfunCloudLive.h"（登录相关方法）

```
#import <Foundation/Foundation.h>

@interface TalkfunCloudLive : NSObject

+ (TalkfunCloudLive *)shared;

//获取user information(已经登录才有)
- (NSDictionary *)getUser;

- (NSString *)getUserFolder;

//获取ID(登录过才有)
- (NSString *)getID;

//是否是登录状态
- (BOOL)isLogin;

//根据用户名和密码登录，callback返回登录信息
//(返回的@"code"为0即登录成功。若不为0，返回的@"msg"会包含错误的信息)
- (void)login:(NSString *)username password:(NSString *)password callback:(void (^)(id re
sult))callback;

//根据token自动登录
- (void)autoLogin:(NSString *)token callback:(void (^)(id result))callback;

//退出登录
- (void)logout:(void (^)(id result))callback;

@end
```

登录的方法以及返回的数据

```
- (void)login:(NSString *)username password:(NSString *)password callback:(void (^)(id re
sult))callback;
- (void)autoLogin:(NSString *)token callback:(void (^)(id result))callback;
```

- 登录成功返回data格式数据：

字段	类型	描述
bid	NSString	主播ID
nickname	NSString	主播名称

- 登录失败返回的数据

字段	类型	描述
code	NSNumber	请求状态码
msg	NSString	错误信息

"TalkfunCourse.h"（课程相关方法）

```
#import <Foundation/Foundation.h>

//课程状态
typedef NS_ENUM(NSInteger, TalkfunCourseStatus) {

    TalkfunCourseStatusNotInTime = 1,
    TalkfunCourseStatusInTime = 2,
    TalkfunCourseStatusEnd = 3
};

@interface TalkfunCourse : NSObject

//获取课程列表
- (void)getCourseList:(void (^)(id result))callback;

//添加课程
- (void)addCourse:(NSString *)courseName startTime:(NSString *)startTime endTime:(NSStrin
g *)endTime callback:(void (^)(id result))callback;

//验证账号是否在直播
- (void)verifyLivingOrNot:(void (^)(id result))callback;

@end
```

获取课程列表的返回信息

```
- (void)getCourseList:(void (^)(id result))callback;
```

- @"total"的格式数据：

字段	类型	描述
total	NSNumber	课程总数

- @"data"的格式数据：

字段	类型	描述
course_id	NSString	课程ID
course_name	NSString	课程名称
start_time	NSString	开始时间(格式：08-21 15:00)
end_time	NSString	结束时间(格式：08-22 17:00)
status	NSNumber	状态：1为未到直播时间，2为在直播时间段内，3为已过直播时间

添加课程的返回信息

```
- (void)addCourse:(NSString *)courseName startTime:(NSString *)startTime endTime:(NSStrin
g *)endTime callback:(void (^)(id result))callback;
```

- 返回code格式说明：0为成功，其他值为失败

验证账号是否在直播的返回信息

```
- (void)verifyLivingOrNot:(void (^)(id result))callback;
```

- 返回的数据说明：

字段	类型	描述
code	string	返回码，code=0为成功，可以进入课程，code=204为该账号正在直播中
msg	string	执行结果信息

"TalkfunSetting.h"（设置相关方法）

```
#import <Foundation/Foundation.h>

@interface TalkfunSetting : NSObject

/*单例*/
+ (id)shared;

/**清除缓存*/
- (void)clearCache;
/**获取缓存大小*/
- (NSInteger)getCacheSize;


/**获取 美颜开关 状态*/
- (BOOL)getBeauty;
/**设置 美颜开关 状态*/
- (void)setBeauty:(BOOL)isOn;



/**获取 涂鸦开关 状态*/
- (BOOL)getGraffiti;
/** 设置 涂鸦开关 状态*/
- (void)setGraffiti:(BOOL)isOn;


/**获取 自动上传开关 状态*/
- (BOOL)getAutoUpload;
/** 设置 自动上传开关  状态*/
- (void)setAutoUpload:(BOOL)isOn;

@end
```

"TalkfunWhiteboard.h"（画板相关方法）

```
//
//  TalkfunWhiteboardV2.h
//  CloudLive
//
//  Created by LuoLiuyou on 16/8/18.
//  Copyright © 2016年 Talkfun. All rights reserved.
//

#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
//观看端 才设置 ,直播端不用设置

typedef enum: NSUInteger{
    TalkfunWhiteboarModelEdit = 0,            //编辑模式可以涂鸦涂鸦
    TalkfunWhiteboarModelScroll = 1,         ///滚动模式不可涂鸦
}TalkfunWhiteboarModel;



typedef enum: NSUInteger{
    TalkfunDrawTypeStroke,//画笔
    TalkfunDrawTypeCircle,//圆形
    TalkfunDrawTypeRectangle,//矩形
    TalkfunDrawTypeStraightLine,//直线
    TalkfunDrawTypeDottedLine,//虚线
    TalkfunDrawTypeBoxDelete,//框删除
    TalkfunDrawTypeArrow,//箭头
    TalkfunDrawTypeText//文本
}TalkfunDrawType;
//画笔颜色
typedef enum: NSUInteger{

    TalkfunDrawBlackColor,  //黑
    TalkfunDrawRedColor,    //红
    TalkfunDrawYellowColor, //黄
    TalkfunDrawGreenColor,  //绿
    TalkfunDrawBlueColor,   //蓝
    TalkfunDrawBrownColor ,  //棕
    TalkfunDrawWhiteColor  , //白
    TalkfunDrawWhiteCyanColor  , //青色
    TalkfunDrawWhitePurpleColor  //紫色
}TalkfunDrawColor;

//画笔大小
typedef enum: NSUInteger{
    TalkfunDrawLineWidthSmall = 2,          //小
    TalkfunDrawLineWidthMiddle = 4,         //中
    TalkfunDrawLineWidthBig = 6,           //大
    TalkfunDrawLineWidthLarge = 8,
```

```
    TalkfunDrawLineWidthHuge = 10            // 特大

}TalkfunDrawLineWidth;

//文字大小
typedef enum: NSUInteger{

    TalkfunDrawTextSizeSmall = 20,          //小
    TalkfunDrawTextSizeMiddle = 25,         //中
    TalkfunDrawTextSizeBig = 30,           //大
    TalkfunDrawTextSizeHuge = 35           //特大
}TalkfunDrawTextSize;




@protocol TalkfunWhiteboardDelegate <NSObject>
@optional

//- (void)whiteboardDidTouched;

//偏移量     是否显示slide    画板内容大小
- (void)witeboardDocument:(CGFloat)contentOffset  contentHeight:( CGFloat )contentHeight
contentSlider:(BOOL)show;
//
//是否还能 撤退;
- (void)whiteboardDidUndo:(BOOL)canUndo;
//是否还能 前进
- (void)whiteboardDidRedo:(BOOL)canRedo;
//请求当前页数 是否有 笔画数据
- (void)requestDrawingBoardData:(NSDictionary*)dict  callback:(void (^)(NSArray *result))
callback;


//观看端:叫代理 对象从本地获取 图片, 然后通过 返回
- (UIImage*)getImageWithUrl:(NSString*)URL;

@end


@interface TalkfunWhiteboard : UIView

//是否支持滚动ppt
//@property (nonatomic,assign)  BOOL scroll ;
@property (nonatomic,assign)  TalkfunWhiteboarModel whiteboarModel;

//是否设置涂鸦  默认 为 NO
@property (nonatomic,assign) BOOL graffiti;


/** 代理对象 */
@property (nonatomic, weak) id<TalkfunWhiteboardDelegate> delegate;
/** 新建画板的背景色 */
```

```
@property(nonatomic,strong)UIColor *whiteboardBackgroundColor;
//画板类型
@property (nonatomic, assign) TalkfunDrawType drawType;



//@property (nonatomic, assign) TalkfunWhiteboarModel WhiteboarModel;

//画笔颜色
@property (nonatomic, assign) TalkfunDrawColor lineColor;
//画笔宽度
@property (nonatomic, assign) TalkfunDrawLineWidth lineWidth;

//文字大小
@property (nonatomic, assign) TalkfunDrawTextSize  textSize;

//当前页码
@property (readonly,nonatomic) NSInteger currentPage;
//当前子页码
@property (readonly,nonatomic) NSInteger currentSubPage;
//总页数
@property (readonly,nonatomic) NSInteger totalPage;

//当前索引
@property (readonly,nonatomic) NSInteger currentIndex;

//当前子页索引
@property (readonly,nonatomic) NSInteger currentSubIndex;



+ (id)shared;
//外面传入rgb颜色
- (void)setLineRgbColor:(UIColor*)rgbColor;

/**上一页*/
- (void)movePrevious:(void (^)(NSDictionary *result))callback;

/**下一页*/
- (void)moveNext:(void (^)(NSDictionary *result))callback;

/**移动到某个索引*/
- (void)moveToIndex:(NSInteger)index callback:(void (^)(NSDictionary *result))callback;



/**插入白板,根据当前的索引,再添加在后面  */
-(void)insertWhiteboard;

/**添加白板, 白板永远在前面,文档在后面 */
-(void)addWhiteboard;
//
```

```
/**撤退*/
- (void)undo;
/**前进*/
- (void)redo;

//是否能够撤退*/
- (BOOL)canUndo;
/**是否能够前进*/
- (BOOL)canRedo;

/**使用偏移量进行文档预览*/
- (void)previewDocumentWithContentOffset:(CGFloat)offset;

/**移动文档偏移量*/
- (void)moveDocumentToContentOffset:(CGFloat)offset;


//传入指令  执行操作
- (void)execute:(NSDictionary *)command;

//清空当前页涂鸦
-(void)clearDraw;
//清除所有页码的涂鸦
-(void)ClearData;

/**销毁*/
- (void)shutdown;
@end




@interface TalkfunWhiteboard : UIView




//是否设置涂鸦  默认为YES
@property (nonatomic,assign) BOOL graffiti;


/** 代理对象 */
@property (nonatomic, weak) id<TalkfunWhiteboardDelegate> delegate;
/** 新建画板的背景色 */
@property(nonatomic,strong)UIColor *whiteboardBackgroundColor;
//画板类型
@property (nonatomic, assign) TalkfunDrawType drawType;



//画笔颜色
@property (nonatomic, assign) TalkfunDrawColor lineColor;
//画笔宽度
```

```
@property (nonatomic, assign) TalkfunDrawLineWidth lineWidth;

//文字大小
@property (nonatomic, assign) TalkfunDrawTextSize  textSize;

//当前页码
@property (readonly,nonatomic) NSInteger currentPage;
//当前子页码
@property (readonly,nonatomic) NSInteger currentSubPage;
//总页数
@property (readonly,nonatomic) NSInteger totalPage;

//当前索引
@property (readonly,nonatomic) NSInteger currentIndex;

//当前子页索引
@property (readonly,nonatomic) NSInteger currentSubIndex;

+ (id)shared;

/**上一页*/
- (void)movePrevious:(void (^)(NSDictionary *result))callback;

/**下一页*/
- (void)moveNext:(void (^)(NSDictionary *result))callback;

/**移动到某个索引*/
- (void)moveToIndex:(NSInteger)index callback:(void (^)(NSDictionary *result))callback;

/**添加白板*/
-(void)addWhiteboard;

/**撤退*/
- (void)undo;
/**前进*/
- (void)redo;

//是否能够撤退*/
- (BOOL)canUndo;
/**是否能够前进*/
- (BOOL)canRedo;

/**使用偏移量进行文档预览*/
- (void)previewDocumentWithContentOffset:(CGFloat)offset;

/**移动文档偏移量*/
- (void)moveDocumentToContentOffset:(CGFloat)offset;


/**销毁*/
- (void)shutdown;

//传入指令  执行操作
```

```
- (void)execute:(NSDictionary *)command;


//清空当前页涂鸦
-(void)clearDraw;
@end
```

画板数据广播

- 广播事件名 （TALKFUN_NOTIFICATION_WHITEBOARD_RELOAD）

参数	类型	描述
thumbnails	NSArray	各个ppt的缩略图地址和它的其它属性
currentIndex	NSNumber	当前索引
currentPage	NSNumber	当前页码
totalPage	NSNumber	总页码

thumbnails数据：

参数	类型	描述
index	NSNumber	ppt索引
thumbnail	NSString	缩略图地址
page	NSNumber	当前页码
backgroundColor	UIColor	背景颜色

```
示例数据：
{
    currentIndex = 0;
    currentPage = 1;
    thumbnails =     (
                {
            backgroundColor = "";
            index = 0;
            page = 1;
            thumbnail = "http://lp2.talk-fun.com/doc/fc/e9/5c/70fa3979944b13787180c1b9ee/
1_1_s.jpg";
        },
                {
            backgroundColor = "";
            index = 1;
            page = 2;
            thumbnail = "http://lp2.talk-fun.com/doc/fc/e9/5c/70fa3979944b13787180c1b9ee/
2_1_s.jpg";
        },
                {
            backgroundColor = "";
            index = 2;
            page = 3;
            thumbnail = "http://lp2.talk-fun.com/doc/fc/e9/5c/70fa3979944b13787180c1b9ee/
3_1_s.jpg";
        }
    );
    totalPage = 10;
}
```

翻页callback返回数据

```
//上一页（翻页，上一页点击按钮调用）
- (void)movePrevious:(void (^)(NSDictionary *result))callback;

//下一页（翻页，下一页点击按钮调用）
- (void)moveNext:(void (^)(NSDictionary *result))callback;

//移动到某个索引（翻页，快速显示指定的某一页）
- (void)moveToIndex:(NSInteger)index callback:(void (^)(NSDictionary *result))callback;
```

参数	类型	描述
currentIndex	NSNumber	当前索引
currentSubIndex	NSNumber	当前子页索引
currentPage	NSNumber	当前页码
currentSubPage	NSNumber	当前子页码
totalPage	NSNumber	总页码

```
示例数据：
{
    currentIndex = 0;
    currentPage = 1;
    currentSubIndex = 1;
    currentSubPage = 2;
    totalPage = 27;
}
```

"TalkfunDocument.h"（文档相关方法）

```
#import <Foundation/Foundation.h>

@interface TalkfunDocument : NSObject

//根据courseID获取PPT文件列表
- (void)getDocumentListOfCourse:(NSString *)courseID callback:(void (^)(id result))callba
ck;

//根据PPT课件ID获取相应课件信息
- (void)getDocument:(NSString *)courseWareID callback:(void (^)(id result))callback;

//根据PPT课件ID加载课件
- (void)loadDocument:(NSString *)courseWareID callback:(void (^)(id result))callback;

//提供courseID和文件（本地图片的asset（PHAsset或者ALAsset类型）的对象的集合 或 文件的地址） 上传图
片或文档，返回上传结果
- (void)upload:(NSString *)courseID files:(NSArray *)files callback:(void (^)(id result))
callback;

//轮询文件处理进度
- (void)processProgress:(NSString *)courseID callback:(void (^)(id result))callback;
//取消上传某个文档
- (void)cancelUpload:(NSString *)courseID filePath:(NSString *)filePath callback:(void (^
)(id result))callback;


@end
```

根据courseID获取PPT文件列表的返回数据

```
- (void)getDocumentListOfCourse:(NSString *)courseID callback:(void (^)(id result))callba
ck;
```

- 返回data格式数据：

字段	类型	描述
id	string	文档ID
name	string	文档名
thumbnail	string	封面缩略图地址

根据PPT课件ID获取相应课件的返回数据

```
- (void)getDocument:(NSString *)courseWareID callback:(void (^)(id result))callback;
```

- 返回data格式数据：

字段	类型	描述
id	string	文档ID
name	string	文档名
url	string	图片url前缀
thumbnail	string	封面缩略图地址
pictures	array	page：页数，title：标题，urls：大图，thumbnailUrls：缩略图
size	int	文档大小，单位为字节

图片上传和文档上传

```
- (void)upload:(NSString *)course_id files:(NSArray *)files callback:(void (^)(id result)
)callback;
```

- 提供courseID和文件（本地图片的asset（PHAsset或者ALAsset类型）的对象的集合 或 文件的地址） 上传图片或文档，返回上传结果

上传事件（广播）

- (上传进度)TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_PROGRESS
- (上传完成)TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_DONE
- (上传失败)TALKFUN_NOTIFICATION_DOCUMENT_UPLOAD_FAIL

轮询文件处理进度

- (void)processProgress:(NSString *)ID callback:(void (^)(id result))callback;
- 未上传过的文件，文件上传完毕返回的数据里面会有个文档的ID，要查看文件处理的进度就根据这个文件的ID查看进度。
- 返回responseObject里面的data格式数据：

字段	类型	描述
bid	string	主播ID
client_ip	string	客户IP
from	string	上传方式
id	string	文件ID
location	string	1代表在云服务器端
course_id	string	绑定的课程ID（没有，为0）
ext	string	文件后缀名
filemd5	string	文件MD5
name	string	文件名
partner_id	string	合作方ID
pid	string	合作方ID
sid	string	存储服务器ID
size	string	文件大小
t	string	文件上传的时间戳
type	string	文件类型

```
//成功上传返回的数据：
{
    code = 0;
    responseObject =     {
        code = 0;
        data =         {
            bid = 12526;
            "client_ip" = "119.130.206.173";
            "course_id" = 0;
            ext = doc;
            filemd5 = dad9e6ce43214270733d1fe37621a6e3;
            from = 1;
            id = 117038;
            location = 1;
            md5 = dad9e6ce43214270733d1fe37621a6e3;
            name = "\U50bb\U903c\U5fc314";
            "partner_id" = 20;
            pid = 20;
            sid = 2;
            sign = 92c29898fc4e345f3db3f538299a404c;
            size = 1575424;
            t = 1488196597;
            type = 2;
        };
        timestamp = 1488196606;
    };
    type = document;
}
```

- 返回document里面的数据：

字段	类型	描述
bid	string	主播ID
course_id	string	绑定的课程ID（没有，为0）
ext	string	文件后缀名
filemd5	string	文件MD5
name	string	文件名
origin	string	图片地址
pages	string	页数
partner_id	string	合作方ID
sid	string	存储服务器ID
size	string	文件大小
thumb	string	文件缩略图地址
type	string	文件类型
url	string	文件地址

已经上传过的文档返回的信息

```
{
    code = "-10";
    document =     {
        bid = 12526;
        "course_id" = 0;
        ext = doc;
        from = 1;
        id = 117038;
        location = 1;
        md5 = dad9e6ce43214270733d1fe37621a6e3;
        name = "\U50bb\U903c\U5fc314";
        origin = "";
        pages = 2;
        "partner_id" = 20;
        sid = 2;
        size = 1575424;
        thumb = "https://lp2-4.talk-fun.com/doc/da/d9/e6/ce43214270733d1fe37621a6e3/thumb
.jpg";
        type = 2;
        url = "https://lp2-4.talk-fun.com/doc/da/d9/e6/ce43214270733d1fe37621a6e3";
    };
    msg = "该文档已经上传过";
}
```

轮询文件处理进度

- (void)processProgress:(NSString *)ID callback:(void (^)(id result))callback;
- 取消上传某个文档

文档上传详情

"TalkfunUtils.h"（工具类）

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>


@interface TalkfunUtils : NSObject

//获取字符串的MD5
+ (nullable NSString *)md5:(nullable NSString *)string;

//获取data的MD5
+ (nullable NSString*)getMd5_32Bit_Data:(nullable NSData*)data;

//获取时间戳
+ (NSInteger)getTimestamp;

//urlEncode
+ (nullable NSString *)urlencode:(nullable NSString *)input;

//获取手机UUID
+ (nullable NSString *)UUID;

//字典转为字符串
+ (nullable NSString *)jsonEncode:(nullable NSDictionary *)input;

//根据提供的字符串、最大size和字符串字体大小获取CGRect
+ (CGRect)getRectWithString:(nullable NSString *)string size:(CGSize)size fontSize:(CGFlo
at)fontSize;

//根据提供的字符串、最大size和字符串字体大小获取CGRect(传入属性值)
+ (CGRect)getRectWithString:(nullable NSString *)string size:(CGSize)size fontSize:(CGFlo
at)fontSize attributes:(nullable NSDictionary<NSString *, id> *)attributes;
//获取每行的文字
+ (nullable NSArray *)getSeparatedLinesFromtext:(nullable NSString *)text font:(nullable
UIFont *)font maxWidth:(CGFloat)maxWidth;
//获取有大图的额外高度
+ (CGFloat)getExtraHeightWithString:(nullable NSString *)string fontSize:(CGFloat)fontSiz
e boundingSize:(CGSize)size row:(NSInteger)row;

//图文混排
+ (nullable NSDictionary *)assembleAttributeString:(nullable NSString *)string boundingSi
ze:(CGSize)size fontSize:(CGFloat)fontSize shadow:(BOOL)shadow;

//提供字节数计算出文件大小
+ (nullable NSString *)fileSizeWithInterge:(NSInteger)size;

//图片修正方向
+ (nullable UIImage *)fixOrientation:(nullable UIImage *)aImage;

//图片压缩
```

```
+ (nullable UIImage*)imageWithImage:(nullable UIImage*)image;

//获取图片大小
+ (CGFloat)getSize:(nullable UIImage *)image;

//获取userAgent
+ (NSString *_Nullable)getUserAgent;

//获取头像urlString
+ (nullable NSString *)getAvatarString:(nonnull NSString *)xid avatarHost:(nonnull NSStri
ng *)avatarHost;

//创建路径
+ (BOOL)mkdirs:(NSString *_Nonnull)directory;

//将数据存档
+ (BOOL)archivedToFile:(NSString *_Nonnull)file data:(id _Nonnull )data;

@end
```

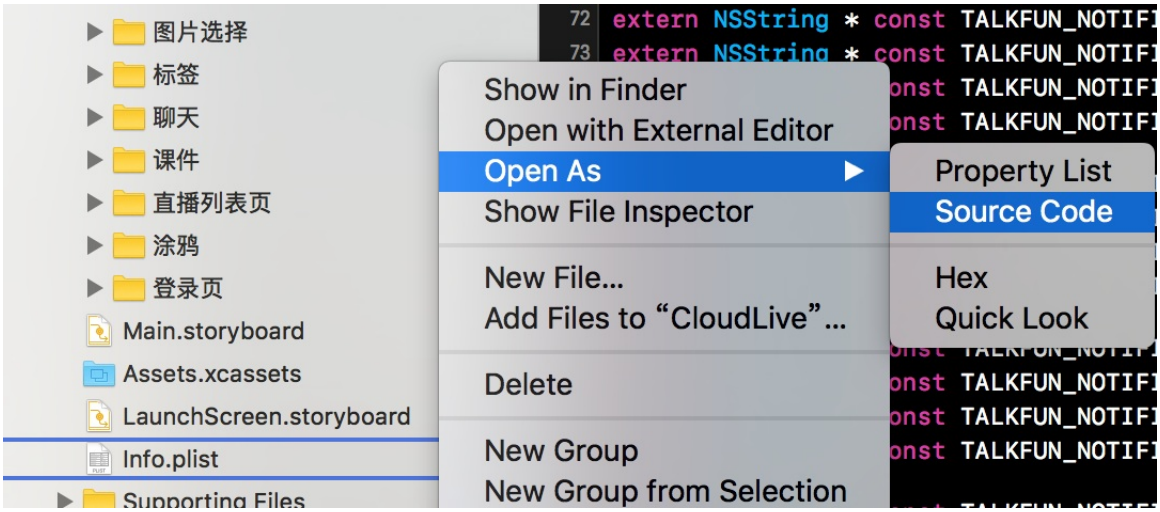
"TalkfunPhotoAssets.h"（照片工具类）

```
#import <Foundation/Foundation.h>
#import <Photos/Photos.h>

@interface TalkfunPhotoAssets : NSObject

//获取图片的标识（NSString类型）
- (NSMutableArray *)getImageIDs:(NSArray *)assetArray;

//根据asset获取原图
- (UIImage *)originalImage:(PHAsset *)asset;

//根据asset获取缩略图
- (UIImage *)thumbnailImage:(PHAsset *)asset sizeCoefficient:(CGFloat)sizeCoefficient;

//获取相册全部asset（PHAsset类型）
- (NSMutableArray *)getImagesAsset;

//获得所有相簿 (PHAssetCollection类型)
- (NSArray *)getAllAlbum;

//根据相簿获取对应的asset数组（PHAsset类型）
- (NSArray *)getPhotoAssets:(PHAssetCollection *)assetCollection;

//根据asset（PHAsset类型）的集合获取缩略图（PHAsset类型）
- (NSMutableArray *)getThumbnailsWithAssetArray:(NSMutableArray *)assetArray;

//根据asset数组获取对应的原图（UIImage类型）
- (NSMutableArray *)getOriginalImagesWithAssets:(NSArray *)assetsArray;

@end
```

文档上传说明

`@文档上传接入`

- 1.设置info.plist文件
  - 以Source Code方式打开info.plist文件



，在里面加入这段代码

```
<key>CFBundleDocumentTypes</key>
<array>
<dict>
   <key>CFBundleTypeExtensions</key>
   <array>
       <string>pptx</string>
   </array>
   <key>CFBundleTypeIconFiles</key>
   <array>
       <string>icon@2x.png</string>
       <string>icon@3x.png</string>
   </array>
   <key>CFBundleTypeName</key>
   <string>PPT Document</string>
   <key>LSHandlerRank</key>
   <string>Default</string>
   <key>LSItemContentTypes</key>
   <array>
       <string>com.microsoft.powerpoint. ppt</string>
   </array>
</dict>
<dict>
   <key>CFBundleTypeExtensions</key>
   <array>
       <string>doc</string>
       <string>docx</string>
   </array>
   <key>CFBundleTypeIconFiles</key>
   <array>
```

```
        <string>icon@2x.png</string>
        <string>icon@3x.png</string>
    </array>
    <key>CFBundleTypeName</key>
    <string>DOC Document</string>
    <key>LSHandlerRank</key>
    <string>Default</string>
    <key>LSItemContentTypes</key>
    <array>
        <string>com.microsoft.word.doc</string>
    </array>
</dict>
<dict>
    <key>CFBundleTypeExtensions</key>
    <array>
        <string>pdf</string>
    </array>
    <key>CFBundleTypeIconFiles</key>
    <array>
        <string>icon@2x.png</string>
        <string>icon@3x.png</string>
    </array>
    <key>CFBundleTypeName</key>
    <string>PDF Document</string>
    <key>LSHandlerRank</key>
    <string>Default</string>
    <key>LSItemContentTypes</key>
    <array>
        <string>com.microsoft.powerpoint.ppt</string>
        <string>public.item</string>
        <string>com.microsoft.word.doc</string>
        <string>com.adobe.pdf</string>
        <string>com.microsoft.excel.xls</string>
        <string>public.image</string>
        <string>public.content</string>
        <string>public.composite-content</string>
        <string>public.archive</string>
        <string>public.audio</string>
        <string>public.movie</string>
        <string>public.text</string>
        <string>public.data</string>
    </array>
</dict>
</array>
```

- 2.在appDelegate.m文件里面引入头文件**#import "TalkfunImportDocumentViewController.h"**,添加以下方法

```
-(BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplicatio
n:(NSString *)sourceApplication annotation:(id)annotation {
if (![self.login isLogin]) {
   [[NSFileManager defaultManager] removeItemAtURL:url error:nil];
   UIAlertView * alert = [[UIAlertView alloc] initWithTitle:@"提示" message:@"未登录不
能上传文件" delegate:self cancelButtonTitle:@"确定" otherButtonTitles:nil, nil];
   [alert show];
   return NO;
}
if (url != nil && [url isFileURL]) {
   UIStoryboard * storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];

   TalkfunImportDocumentViewController * importVC = [storyboard instantiateViewContro
llerWithIdentifier:@"importController"];

   TalkfunLoginNavigationController * nav = (TalkfunLoginNavigationController *)self.
window.rootViewController;
   if ([nav.viewControllers.lastObject isKindOfClass:[TalkfunImportDocumentViewContro
ller class]]) {
       //            [self.window.rootViewController presentViewController:importVC a
nimated:YES completion:nil];
       importVC = nav.viewControllers.lastObject;
       [importVC addFile:url];
   }else if(![nav.viewControllers.lastObject isKindOfClass:[TalkfunLiveController cla
ss]]) {

       [nav pushViewController:importVC animated:YES];
       [importVC addFile:url];
   }else if ([nav.viewControllers.lastObject isKindOfClass:[TalkfunLiveController cla
ss]]){
       TalkfunLiveController * liveVC = nav.viewControllers.lastObject;
       [liveVC uploadFile:url];
   }
}
return YES;
}
```

- 3.引入Module/UploadFile文件夹的全部东西

Finish

`@上传说明`

- 状态码

```
TalkfunCloudLiveCodeSuccess                = 0         成功
TalkfunCloudLiveCodeUploading              = 500     文件正在上传
TalkfunCloudLiveCodeHasBeenUploaded          = 501     文件已经上传过
TalkfunCloudLiveCodeInTheProcessing      = 33     文件处理中
TalkfunCloudLiveCodeProcessFial             = 34      文件处理失败
TalkfunCloudLiveCodeUploadFail             = 503     文件上传失败
```

- 上传调用
  - 调用TalkfunDocument.h的以下接口，如果上传到当前帐号，courseID参数传入"0"，如果想传入到相应的课程则传入课程的ID，files参数是传入文档的路径集合。

```
//提供courseID和文件（本地图片的asset（PHAsset或者ALAsset类型）的对象的集合 或 文件的地址） 上传图片或文档，返回上传结果
- (void)upload:(NSString *)courseID files:(NSArray *)files callback:(void (^)(id result))callback;
```

---

callback返回的数据：

- 文档上传中

```
{
    code = 500;
    progress = "<NSProgress: 0x17013ebe0> : Parent: 0x0 / Fraction completed: 0.0208 / Completed: 32768 of 1576114  ";
    type = document;
}
```

- 文档上传完成

```
{
    code = 0;
    responseObject =     {
        code = 0;
        data =         {
            bid = 12526;
            "client_ip" = "219.136.204.204";
            "course_id" = 0;
            ext = doc;
            filemd5 = 531561e09d6101b2f5cb11dacead5c86;
            from = 1;
            id = 129484;
            location = 1;
            md5 = 531561e09d6101b2f5cb11dacead5c86;
            name = "\U50bb.\U903c.\U5fc3.31";
            "partner_id" = 20;
            pid = 20;
            sid = 2;
            sign = 4673b7e5beb39b9627074fad666abdfd;
            size = 1575936;
            t = 1491031650;
            type = 2;
        };
        timestamp = 1491031656;
    };
    type = document;
}
```

- 文档已经上传过

```
{
code = 501;
msg = "\U8be5\U6587\U6863\U5df2\U7ecf\U4e0a\U4f20\U8fc7";
responseObject =     {
  bid = 12526;
  "course_id" = 0;
  ext = doc;
  from = 1;
  id = 129485;
  location = 1;
  md5 = 19c9bd8d24daaeaeafa608e7adcfc9ea;
  name = "\U50bb.\U903c.\U5fc3.32";
  origin = "";
  pages = 2;
  "partner_id" = 20;
  sid = 2;
  size = 1575424;
  thumb = "https://lp2-4.talk-fun.com/doc/19/c9/bd/8d24daaeaeafa608e7adcfc9ea/thumb.j
pg";
  type = 2;
  url = "https://lp2-4.talk-fun.com/doc/19/c9/bd/8d24daaeaeafa608e7adcfc9ea";
};
type = document;
}
```

- 文档上传失败

```
{
    @"code":503;
    @"msg":error,@"type":@"document"
}
```

- 文档处理中

```
{
    code = 33;
    data =     {
        percent = "83.7";
    };
    msg = "\U6587\U6863\U6b63\U5728\U5904\U7406\U4e2d";
}
```

- 文档处理成功

```
{
    cache = 1;
    code = 0;
    data =     {
        bid = 12526;
        "convert_server" = 3232236136;
        "course_id" = 0;
        donetime = 1491031672;
        ext = doc;
        from = 1;
        id = 129484;
        images = 2;
        info = "";
        location = 1;
        md5 = 531561e09d6101b2f5cb11dacead5c86;
        name = "\U50bb.\U903c.\U5fc3.31";
        pages = 2;
        pid = 20;
        roomid = 550481;
        sid = 2;
        size = 1575936;
        status = 0;
        time = 1491031656;
        type = 2;
        url = "http://lp2-4.talk-fun.com/doc/53/15/61/e09d6101b2f5cb11dacead5c86";
        urlLocal = "http://lp2-4.talk-fun.com/doc/53/15/61/e09d6101b2f5cb11dacead5c86";
    };
}
```

- 文档处理失败

```
{
    code = 34;
    msg = "\U6587\U6863\U6b63\U5728\U5904\U7406\U4e2d";
}
```

SDK和文档下载

- Version 2.9.9 (2021-11-20)
  - 直播线路优化
  - SDK下载
- Version 2.9.7 (2021-11-16)
  - 课件上传优化
  - 批量删除涂鸦优化
- Version 2.9.3 (2021-10-21)
  - iOS15新特性兼容适配
- Version 2.9.1 (2021-10-8)
  - iOS15新特性兼容适配
  - 升级第三方UI库
- Version 2.8.9 (2021-9-15)
  - 解决一些已知问题
- Version 2.8.7 (2021-8-27)
  - 大班、生活直播支持聊天@回复、删除指定聊天内容
  - 小班直播一对一、一对六、一对多支持课件模式、主讲模式、画廊模式切换
  - 新表情添加鲜花表情
  - 小班直播支持未连麦获取涂鸦权限
  - 小班支持新版后台一些配置设置
- Version 2.8.4 (2021-6-22)
  - 优化虚拟机器人聊天信息显示头像
  - 优化大班回放文件列表显示
- Version 2.8.2 (2021-6-22)
  - 优化虚拟机器人聊天信息显示头像
  - 优化大班回放文件列表显示
- Version 2.8.0 (2021-5-27)
  - 生活直播优化
- Version 2.7.6 (2021-4-29)
  - 生活直播优化
- Version 2.7.4 (2021-3-2)
  - UI体验优化
- Version 2.7.3 (2021-1-14)
  - 支持创建一对一、一对六课程类型
  - 支持一对一、一对六直播
  - 优化若干体验问题

- Version 2.7.1 (2020-12-30)
    - 支持手机号码登录
- Version 2.7.0.1 (2020-12-24)
    - 性能优化
- Version 2.7.0 (2020-12-18)
    - 直播器线程优化
    - 修复已知问题
- Version 2.6.9 (2020-12-8)
    - 直播器录制回放与镜像优化
    - 修复已知问题
- Version 2.6.7 (2020-11-23)
    - 小班直播间改版
    - 修复已知问题
- Version 2.6.6 (2020-10-27)
    - 直播超时响应后台结束直播指令
    - 修复已知问题
- Version 2.6.3 (2020-10-02)
    - IOS 14 适配
    - 修复已知问题
- Version 2.5.9 (2020-07-02)
    - 支持腾讯rtc直播
    - 直播的默认线路优化
- Version 2.5.8 (2020-06-18)
    - 支持踢主播下线
    - 生活直播新添镜像特效)
- Version 2.5.7 (2020-06-12)
    - 生活直播UI显示优化
    - 生活直播新添镜像特效
- Version 2.5.19 (2020-5-19)
    - 生活直播支持开关美颜
- Version 2.5.5 (2020-4-30)
    - 直播sdk 录制优化
- Version 2.5.3 (2019-4-22)
    - 大班直播器的分辨率显示优化
- Version 2.5.2 (2019-4-15)
    - 大班互动的旁路的优化
    - 去掉webView
- Version 2.4.9 (2019-2-27)

    - 优化iCloud照片的显示与选择
- Version 2.3.5 (2019-2-11)
    - 生活直播切换分辨率
    - 生活直播切换线路
- Version 2.3.4 (2019-9-11)
    - SDK从libCloudLiveSDK.a 替换到 CloudLiveSDKFramework.framework
    - 修复已知bug
- Version 2.3.1 (2019-7-25)
    - 优化画板的文本
    - 修复已知bug
- Version 2.3.0 (2019-5-24)
    - 优化小班
    - 修复已知bug
- Version 1.3.7 (2019-5-10)
    - 班课互动和视频连麦模式功能优化
    - 修复已知bug
- Version 1.3.4 (Date: 2018-08-21)
    - TalkfunLive 模块调用优化
    - 画板性能优化
    - 视频上传处理速度优化
    - 新增T主播下线的通知
- Version 1.3.3 (Date:2017-06-20)
    - 支持自动转屏
    - 支持横屏
    - 扩展emit接口，支持id类型参数
    - 新增事件类型定义
    - 支持暂停、恢复直播
    - 新增用户自定义头像支持
    - 开放白板字体大小、线粗细接口
    - 画板性能优化
    - 纯音频推流关键帧优化
- Version 1.3.2
    - 优化回放上传
    - 修复已知问题
- Version 1.3.1
    - 优化直播推流和视频录制
- Version 1.3.0
    - 回放上传
    - 在线人数列表
    - 文档接收上传

    - 美颜开关
    - 涂鸦开关
    - 涂鸦新增功能
    - ppt区域可滚动
    - 修复已知bug

- Version 1.2.0

    - 新增视频录制
    - 新增视频上传
    - 美颜功能修复
    - 代码重构

- Version 1.1.0

    - 架构优化
    - 修复PPT翻页Bug

- Version 1.0.0

    - 第一个版本
    - 摄像头音视频直播
    - 文档展示