拓奇录播互动模块开发指导文档

文档状态	[ ]初稿	文档标识	
	[ ]通过	当前版本	1.0
	[△]修改	作者	zp
	[ ]发布	公司/部门	研发中心互动录播产品线
	[ ]作废	完成日期	2019 年 09 月 31 日

（内部资料请勿外传）

文档变更记录

序号	变更（+/-）说明	作者	版本号	日期

目录

概述

本文为对接拓奇录播互动模块的开发人员而写，目的是供您了解互动模块与第三方模块进行对接的接口以及流程，方便第三方库开发人员开发出可以直接对接我们录播系统的动态库。

一、互动模块文件描述：

1. libconference_sdk.so: 互动模块动态库
2. conference_sdk.h: 动态库接口头文件

二、互动模块 API 接口描述：

init

【描述】

初始化互动模块

【语法】

ret_code_e init(const audio_mixer_props_t *pmixer_props, const conference_event_handler_t *pevent_handler);

【参数】

参数名称	描述	输入/输出
pmixer_props	混音器输出属性	输入
pevent_handler	事件上报回调函数集	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

uninit

【描述】

反初始化互动模块

【语法】

ret_code_e uninit();

【参数】

无

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

connect_MCU

【描述】

连接服务器

【语法】

```
ret_code_e connect_MCU(const char* pserver_ddr, int server_port, const char* pusername,
                       const char* ppassWord);
```

【参数】

参数名称	描述	输入/输出
pserver_ddr	服务器地址	输入
server_port	服务器端口	输入
pusername	用户名	输入
ppassWord	用户密码	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

disconnect_MCU

【描述】

断开服务器

【语法】

ret_code_e disconnect_MCU();

【参数】

无

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

get_conference_room_count

【描述】

获取服务器上会议室个数

【语法】

ret_code_e get_conference_room_count(int* pcount);

【参数】

参数名称	描述	输入/输出
pcount	缓存会议室个数地址	输出

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

get_conference_room_info

【描述】

获取会议室信息

【语法】

ret_code_e get_conference_room_info(int room_indx, room_info_t* proom_info);

【参数】

参数名称	描述	输入/输出
room_indx	服务器上会议室序号	输入
proom_info	缓存会议室信息	输出

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

login_conference_room

【描述】

登录会议室

【语法】

ret_code_e login_conference_room(int conference_room_id);

【参数】

参数名称	描述	输入/输出
conference_room_id	会议室 ID	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

logout_conference_room

【描述】

登出会议室

【语法】

ret_code_e logout_conference_room();

【参数】

无

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

kickout_conference_room

【描述】

踢出会议室

【语法】

ret_code_e kickout_conference_room(int user_id);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

invite

【描述】

邀请通话（建立点对点也用这个）:请求加入+邀请加入通用，内部逻辑做区分

【语法】

ret_code_e invite(int user_id, intvite_type_e intvite_type, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
intvite_type	通话类型，详见 intvite_type_e 表	输入
ctrl	0-停止邀请/请求 1-开始邀请/请求	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

- 点对点也用这个
- 请求加入 和 邀请加入都用这个，主讲为邀请加入，听讲为请求加入

ctrl_user_invite

【描述】

控制用户的邀请

【语法】

ret_code_e ctrl_user_invite(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-拒绝 1-同意	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

get_own_user_info

【描述】

获取自己的用户信息

【语法】

ret_code_e get_own_user_info(user_info_t* puser_info);

【参数】

参数名称	描述	输入/输出
puser_info	缓存用户信息地址	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

get_user_count

【描述】

获取会议室内用户数

【语法】

ret_code_e get_user_count(int* pcount);

【参数】

参数名称	描述	输入/输出
pcount	缓存用户数	输出

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

get_user_info

【描述】

获取指定 ID 的用户信息

【语法】

ret_code_e get_user_info(int user_id, user_info_t* puser_info);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
puser_info	缓存用户信息地址	输出

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

send_chat

【描述】

发送聊天信息

【语法】

ret_code_e send_chat(const conference_chat_info_t* pconference_chat_info);

【参数】

参数名称	描述	输入/输出
pconference_chat_info	缓存聊天信息地址	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

apply_for_master

【描述】

申请主讲

【语法】

ret_code_e apply_for_master(int ctrl);

【参数】

参数名称	描述	输入/输出
ctrl	0-停止申请 1-开始申请	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

apply_for_talk

【描述】

申请发言

【语法】

ret_code_e apply_for_talk(int ctrl);

【参数】

参数名称	描述	输入/输出
ctrl	0-停止申请 1-开始申请	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

apply_for_aux

【描述】

申请辅流权限

【语法】

ret_code_e apply_for_aux(int ctrl);

【参数】

参数名称	描述	输入/输出
ctrl	0-停止申请 1-开始申请	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

ctrl_user_talk_apply

【描述】

拒绝/允许用户发言 + 停止正在进行的发言

【语法】

ret_code_e ctrl_user_talk_apply(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-拒绝/取消/放弃 1-同意	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

- 拒绝或同意发言申请使用这个接口；
- 取消用户的发言也用这个；
- 放弃发言权限也用这个；

auth_audio

【描述】

授权音频权限

【语法】

ret_code_e auth_audio(int src_user_id, int dst_user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
src_user_id	授权出去的用户	输入
dst_user_id	接受授权的用户	输入
ctrl	0-取消授权 1-授权	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

auth_video

【描述】

授权视频权限

【语法】

ret_code_e auth_video(int src_user_id, int dst_user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
src_user_id	授权出去的用户	输入
dst_user_id	接受授权的用户	输入
ctrl	0-取消授权 1-授权	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

check_user_video

【描述】

查看用户视频：用户向服务器发送流

【语法】

ret_code_e check_user_video(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-取消查看 1-查看	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

check_user_audio

【描述】

查看用户音频：用户向服务器发送流

【语法】

ret_code_e check_user_audio(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-取消查看 1-查看	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

bcast_user_video

【描述】

给房间内所有人授权查看 user_id 的视频

【语法】

ret_code_e bcast_user_video(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-取消授权 1-授权	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

bcast_user_audio

【描述】

给房间内所有人授权查看 user_id 的音频

【语法】

ret_code_e bcast_user_audio(int user_id, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
ctrl	0-取消授权 1-授权	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

set_interact

【描述】

切换互动端

【语法】

ret_code_e set_interact(int user_id);

【参数】

参数名称	描述	输入/输出
user_id	开始互动用户 ID	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

send_stream

【描述】

向服务器发流

【语法】

ret_code_e send_stream(const media_info_t* pmedia_info, const char* pdata, int data_len,
conference_media_type_e media_type);

【参数】

参数名称	描述	输入/输出
pmedia_info	媒体信息	输入
pdata	缓存流数据的地址	输入
data_len	流数据长度	输入
media_type	流类型，详见 conference_media_type_e	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

recv_stream

【描述】

向服务器发送接收流请求

【语法】

ret_code_e recv_stream(int user_id, conference_media_type_e media_type, int ctrl);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
media_type	流类型	输入
ctrl	0-停止接收 1-开始接收	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

send_transparent_data

【描述】

透传上层用户数据接口

【语法】

ret_code_e send_transparent_data(int user_id, const char *pdata, int data_len, int data_type);

【参数】

参数名称	描述	输入/输出
user_id	用户 ID	输入
pdata	用户数据缓存地址	输入
data_len	数据长度	输入
data_type	数据类型 0-二进制 1-字串	输入

【返回值】

retCode_Success	成功
非 retCode_Success	失败，其值为相关错误码

【注意】

无

三、数据类型：

- conference_state_e：定义事件回调状态类型。
- conference_media_type_e：定义媒体类型。
- result_code_e：定义事件回调结果类型。
- ret_code_e：定义接口返回值类型。
- intvite_type_e：定义邀请/请求类型。
- aux_stream_state_info_t：定义辅流状态信息结构体。
- conference_chat_info_t：定义聊天信息结构体。
- media_info_t：定义媒体信息结构体。
- room_info_t：定义房间（会议室）信息结构体。
- user_info_t：定义用户信息结构体。
- conference_event_handler_t：定义事件回调函数结构体。
- audio_mixer_props_t：定义混音输出属性结构体。

conference_state_e

【说明】

定义事件回调状态类型。

【定义】

```
typedef enum _conference_state_e{
        CONFERENCE_STATE_LOGIN_MCU = 1,     // 连接 MCU 获取会议室列表完成
        CONFERENCE_STATE_LOGIN_ROOM,       // 登录指定会议室完成,返回本地用户信息
        CONFERENCE_STATE_USER_ENTER,              // 有用户进入会议室
        CONFERENCE_STATE_USER_LEAVE,              // 有用户离开会议室
        CONFERENCE_STATE_ROOM_EXIT,               // 自己离开会议室
        CONFERENCE_STATE_USER_TALK,               // 有用户音频状态变化
        CONFERENCE_STATE_USER_BCAST,          // 有用户视频状态变化
        CONFERENCE_STATE_USER_CHAIR,          // 有用户主讲状态变化
        CONFERENCE_STATE_LOOKED,              // 作为听讲时，被查看视频
```

```
    CONFERENCE_STATE_USER_CLOSE,          // 连接异常.
    CONFERENCE_STATE_USER_REQTALK,        //用户申请发言
    CONFERENCE_STATE_CONNECT_MCU_FAILED,  //登录 MCU 失败
    CONFERENCE_STATE_LOGIN_ROOM_FAILED,   //登录房间失败
    CONFERENCE_STATE_USER_KICKED,         //用户被踢出了房间
    CONFERENCE_STATE_USER_VPOS,           //用户视频布局位置改变
    CONFERENCE_STATE_UPDATE_INTERACT,     //互动端有更新
    CONFERENCE_STATE_USER_CHAT,           // 收到聊天信息
    CONFERENCE_STATE_USER_VNC,            // VNC 状态变化
    CONFERENCE_STATE_NICK_NAME,           //用户昵称发生变化
    CONFERENCE_STATE_RECV_JOIN,           //收到加入信息
    CONFERENCE_STATE_RECV_INVITE,         //收到邀请信息
    CONFERENCE_STATE_REJECT_JOIN,         //收到拒绝加入信息
    CONFERENCE_STATE_ALLOW_JOIN,          //收到同意加入信息
    CONFERENCE_STATE_REJECT_INVITE,       //收到拒绝邀请信息
    CONFERENCE_STATE_ALLOW_INVITE,        //收到同意邀请信息
    CONFERENCE_STATE_TRANSPARENT,         //透传数据
    CONFERENCE_STATE_USER_AUX,            //有用户辅流状态发生变化

    //CONFERENCE_STATE_USER = 10000,      //用户自定义信息  使用+1 的方式递增
}conference_state_e;
```

conference_media_type_e

【说明】

定义媒体类型。

【定义】

```
typedef enum _conference_media_type_e{
    CONFERENCE_MEDIA_TYPE_AUDIO = 1,//音频流
    CONFERENCE_MEDIA_TYPE_VIDEO_MAJOR,//主流
    CONFERENCE_MEDIA_TYPE_VIDEO_MINOR,//辅流
```

```
        CONFERENCE_MEDIA_TYPE_VIDEO_MINOR2,//辅流 2
}conference_media_type_e;
```

result_code_e

【说明】

定义事件回调结果类型。

【定义】

详见错误码

ret_code_e

【说明】

定义接口返回值类型。

【定义】

详见错误码

intvite_type_e

【说明】

定义邀请/请求类型。

【定义】

```
typedef _intvite_type_e{
        INVITE_TYPE_VIDEO,      //视频会议
        INVITE_TYPE_AUDIO,      //语音会议
}intvite_type_e;
```

aux_stream_state_info_t

【说明】

定义辅流状态信息结构体。

【定义】

```
typedef struct _aux_stream_state_info_s{
```

```
    int m_i32UserID,//用户 id
    int m_i32ResponseState;//响应状态 0-失败  1-成功  2-申请中(暂未使用)
    conference_media_type_e m_eMediaType;//流类型
}aux_stream_state_info_t;
```

conference_chat_info_t

【说明】

定义聊天信息结构体。

【定义】

```
typedef struct _conference_chat_info_s{
    int m_i32SrcID;       //聊天信息来源
    int m_i32DstID;       //发给谁
    int m_i32Color;       //字体颜色
    char m_strContent[128]; //消息内容
}conference_chat_info_t;
```

media_info_t

【说明】

定义媒体信息结构体。

【定义】

```
typedef struct _media_info_s
{
    unsigned short m_u16MediaType;        //媒体类型 0-audio 1-main video 2-aux video...
    unsigned short m_u16CodecID;          //编码类型,视频编码时  3 表示 h264,  音频编码
                                            时, 0-pcm 1-opus    2-g7221    3-amrwb
    unsigned short m_u16FrameSeq;    //低 8 位:组包标识    1-始  0-中间  2-结束  3 完整包
    unsigned short m_u16KeyFrame;         //视频关键帧 0-否  1-是  或者音频采样位数
    unsigned short m_u16Framerate;   //视频帧率或者一个音频编码包的持续时间(单位
                                            ms)
    unsigned short m_u16StreamID;              //数据流序号
    int          m_u32W_S;                      //视频宽  或音频采样率
```

```
    int         m_u32H_C;                    //视频高 或音频声道数
    int         m_u32Bitrate;                //码率,单位 K(原始数据码率=0)
    unsigned long long m_u64TimeStamp;       //时间戳(ms)
}media_info_t;
```

room_info_t

【说明】

定义房间（会议室）信息结构体。

【定义】

```
typedef struct _room_info_s{
    char m_strRoomID[64];        //会议室 ID
    char m_strRoomName[32];      //会议室名称
    char m_strCreatorID[32];     //会议室创建者 ID
    time_t    m_tStartTime;      //会议开始时间
    time_t    m_tEndTime;        //会议开始时间
    int m_s32MaxUserCnt;         //会议室支持的最大成员数
    int m_s32CurUserCnt;         //会议室当前成员数
}room_info_t;
```

user_info_t

【说明】

定义用户信息结构体。

【定义】

```
typedef struct _user_info_s{
    /***user 自身的属性*******/
    unsigned char m_u8IsMaster; //是否主讲 0 - 听讲  1 - 主讲, 暂时不考虑中间态
    unsigned char m_u8IsChairMan;//是否主席
    unsigned char m_u8BcastVideo;//是否广播 video，广播定义为全授权
    unsigned char m_u8BcastAudio;//是否广播 audio，广播定义为全授权
```

```
        unsigned char m_u8GainAux[CONFERENCE_AUX_MAX];//是否获取到辅流资源
        unsigned char m_u8SignalStatus;   //信号状态（强度）

        /***user 在本端的属性*******/
        char m_strID[32];    //用户 ID
        char m_strUserName[32];//用户名
        char m_strNickName[32];//用户昵称
        unsigned char m_u8AuthVideo;//有没有权限拉视频流（是否被授权） 0-不能 1-正在
                                    //申请授权   2-可以
        unsigned char m_u8AuthAudio;//有没有权限拉音频流（是否被授权） 0-不能 1-正在
                                    //申请授权   2-可以
        unsigned char m_u8AudioState;//音频状态,0 未发言 1 请求发言(等待发言) 2 正在
                                    //发言
}user_info_t;
```

conference_event_handler_t

【说明】

定义事件回调函数结构体。

【定义】

```
typedef struct {
        /**  聊天信息上报器**/
        void (*on_conference_chat) (const conference_chat_info_t *pchat_info);
        /**  会议中各种事件上报接口**/
        void (*on_conference_state) (result_code_e result, conference_state_e
            conference_state_id, const user_info_t *puser_info, void *append_data);

    /**  数据流上报接口**/
        void (*on_conference_stream_data) (const media_info_t *pmedia_info, const char
            *pdata, int data_len, int user_id, conference_media_type_e media_type);
} conference_event_handler_t;
```

audio_mixer_props_t

【说明】

定义混音输出属性结构体。

【定义】

```
typedef struct _audio_mixer_props_s{
        int m_i32Sample;//采样率
        int m_i32Channel;//通道数
        int m_i32SampleBit;//每采样数 bit 位
}audio_mixer_props_t;
```

四、错误码：

错误代码	枚举只定义	描述
0	retCode_Success	成功
1	retCode_False	失败
2	retCode_ParamErr	参数错误
3	retCode_InitErr	初始化错误
4	retCode_InitedErr	已经初始化
5	retCode_UninitErr	反初始化错误
6	retCode_SockErr	SOCK 错误或者异常
7	retCode_AddrErr	IP 地址错误
8	retCode_PortErr	端口错误
9	retCode_CreateSockErr	创建 SOCK 错误
10	retCode_SockBindErr	绑定 SOCK 错误
11	retCode_SockCnntErr	连接 SOCK 错误
12	retCode_SockListenErr	监听 SOCK 错误
13	retCode_SockAcceptErr	接收 SOCK 连接错误
14	retCode_SockSendErr	sock 发送数据错误
15	retCode_SockRecvErr	sock 接收数据错误
16	retCode_SizeNotEnough	空间或大小不够
17	retCode_SizeEnough	空间或大小足够了
18	retCode_MsgErr	指令错误
19	retCode_CurlErr	Curl 库错误
20	retCode_CurlInitErr	CURL 库初始化失败
21	retCode_CurlSendErr	CURL 库发送失败
22	retCode_CurlRecvErr	CURL 库接收失败
23	retCode_FileErr	文件错误或异常

24	retCode_FileExistErr	文件已存在
25	retCode_FileNoteExistErr	文件不存在
26	retCode_FileHaveOpen	文件已经打开
27	retCode_FileCreateErr	创建文件错误
28	retCode_FileOpenErr	打开文件错误
29	retCode_FileReadErr	读文件错误
30	retCode_FileReadEnd	读到文件结尾
31	retCode_FileWriteErr	写文件错误
32	retCode_FileSeekErr	文件 seek 错误
33	retCode_FileCloseErr	关闭文件错误
34	retCode_FileCopyErr	复制文件错误
35	retCode_FileCutErr	剪切文件错误
36	retCode_FileFormatErr	文件格式错误
37	retCode_FileSameErr	操作或对象重复
38	retCode_FileAccessErr	文件无法访问
39	retCode_FileSizeErr	大小尺寸不对
40	retCode_FileFindErr	查找失败
41	retCode_FileRemoveErr	删除文件错误
42	retCode_DirErr	目录错误或异常
43	retCode_DirExistErr	目录已存在
44	retCode_DirNotexistErr	目录不存在
45	retCode_DirCreateErr	创建目录错误
46	retCode_DirOpenErr	打开目录错误
47	retCode_DirCloseErr	关闭目录错误
48	retCode_DirCopyErr	复制目录错误
49	retCode_DirCutErr	剪切目录错误
50	retCode_DirFindErr	查找目录错误
51	retCode_DirSameErr	目录重名
52	retCode_DirAccessErr	目录无法访问
53	retCode_DirRemoveErr	删除目录错误
54	retCode_MemoryErr	内存错误或异常
55	retCode_MemoryAllocErr	内存分配错误
56	retCode_MemoryFreeErr	内存释放错误
57	retCode_MemoryOutErr	内存大小不够
58	retCode_StreamErr	流错误(文件流、数据流等)
59	retCode_StreamNotErr	不存在这个流
60	retCode_StreamExistErr	已存在这个流
61	retCode_ListErr	列表错误
62	retCode_QueueErr	队列错误
63	retCode_QueueNodeErr	队列中节点错误
64	retCode_QueueEmptyErr	队列空
65	retCode_QueueFullErr	队列满

66	retCode_QueueFindErr	查找节点失败
67	retCode_XmlErr	xml 错误
68	retCode_XmlOpenErr	打开 xml 文件失败
69	retCode_XmlNodeErr	获取 XML 节点内容失败
70	retCode_XmlAttributeErr	获取 XML 节点属性内容失败
71	retCode_XmlNodeNotExist	XML 节点不存在
72	retCode_RecordErr	录制错误
73	retCode_RecordRunErr	录制进行中
74	retCode_RecordNotStart	录制已结束、或者录制未开始
75	retCode_RecordStopErr	录制停止失败
76	retCode_RecordDircetErr	启动电影模式录制失败
77	retCode_RecordResErr	启动资源模式录制失败
78	retCode_VaCheckErr	va 函数调用失败
79	retCode_VaProfileErr	不支持这种 profile
80	retCode_VaRcErr	不支持这种码率控制模式
81	retCode_VaAdmixing	合成模式中，不支持同时编解码
82	retCode_EncSupportErr	不支持编码
83	retCode_EncErr	编码失败
84	retCode_EncInitedErr	编码器初始化失败
85	retCode_EncNotFindErr	找不到编码器
86	retCode_EncUninitErr	编码器未初始化
87	retCode_ParseErr	解析失败
88	retCode_DecSupportErr	不支持解码
89	retCode_DecSupportYUV420	不支持解码成 yuv420
90	retCode_DecNotFindEntryPoint	找不到入口点
91	retCode_DecFailed	解码失败
92	retCode_OpNotSupport	操作不被支持
93	retCode_YUVFormatErr	不支持的 YUV 格式
94	retCode_IDEnrityErr	身份 ID 错误
95	retCode_Timeout	超时错误
96	retCode_NotObjectErr	找不到对象
97	retCode_ObjectSupportErr	对象不支持此功能
98	retCode_DataErr	数据错误(非法数据)
99	retCode_ZlibErr	zlib 库错误

100	retCode_NotLoginErr	没有登陆
101	retCode_ReloginedErr	不能重复登陆了
102	retCode_StateErr	状态不对
103	retCode_ThreadErr	线程错误
104	retCode_CamCtrlErr	远摇摄像头失败
105	retCode_AccessErr	拒绝访问
106	retCode_UserPasswordErr	用户名或密码错误
107	retCode_NotLoginInfo	未设置登陆信息
108	retCode_NoScreenConnection	没有显示器接入
109	retCode_UnSupportResolution	不支持的分辨率
110	retCode_UnCompleteData	不完整的数据
1000	retCode_Unknow	未知错误
0x1002	retCode_Exception	异常错误
0x1003	retCode_ServerInterErr	服务器内部错误
0x1004	retCode_NvalidSessionErr	无效的会话
0x1005	retCode_NotFindUserErr	找不到用户
0x1006	retCode_VerifiFailureErr	验证失败
0x1007	retCode_DenyAccessErr	拒绝访问
0x1008	retCode_InvalidParam	参数错误
0x1009	retCode_SytemMantainErr	系统正在维护
0x100a	retCode_MuchVisitedErr	访问人数过多
0x100b	retCode_NotFindFileErr	文件未找到
0x100c	retCode_NotFindFileListErr	文件列表未找到
0x100d	retCode_FileListExist	文件列表已存在
0x100e	retCode_FileTransChnUnexist	文件传输通道不存在
0x100f	retCode_FileUploading	文件正在上传中
0x1010	retCode_FileTooLarge	文件大小超过限定值
0x10011	retCode_SenderDidnotUpload	发送者还未上传该文件
0x10012	retCode_FileExist	文件已存在
0x10013	retCode_NanmeOrPasswordErr	用户名或密码错误
0x10014	retCode_RoomUnexist	房间未找到
0x10015	retCode_RoomFull	房间已满
0x10016	retCode_ApplicationTypeMismatch	应用类型不匹配
0x10017	retCode_ServerMismatch	服务器不匹配
0x10018	retCode_SpecifiedServerNotFound	未找到指定的服务器
0x10019	retCode_ServerFull	服务器已满员
0x1001a	retCode_UserExist	用户已存在
0x1001b	retCode_NeedRoomPassword	需要输入房间密码
0x1001c	retCode_RoomLock	房间已锁定

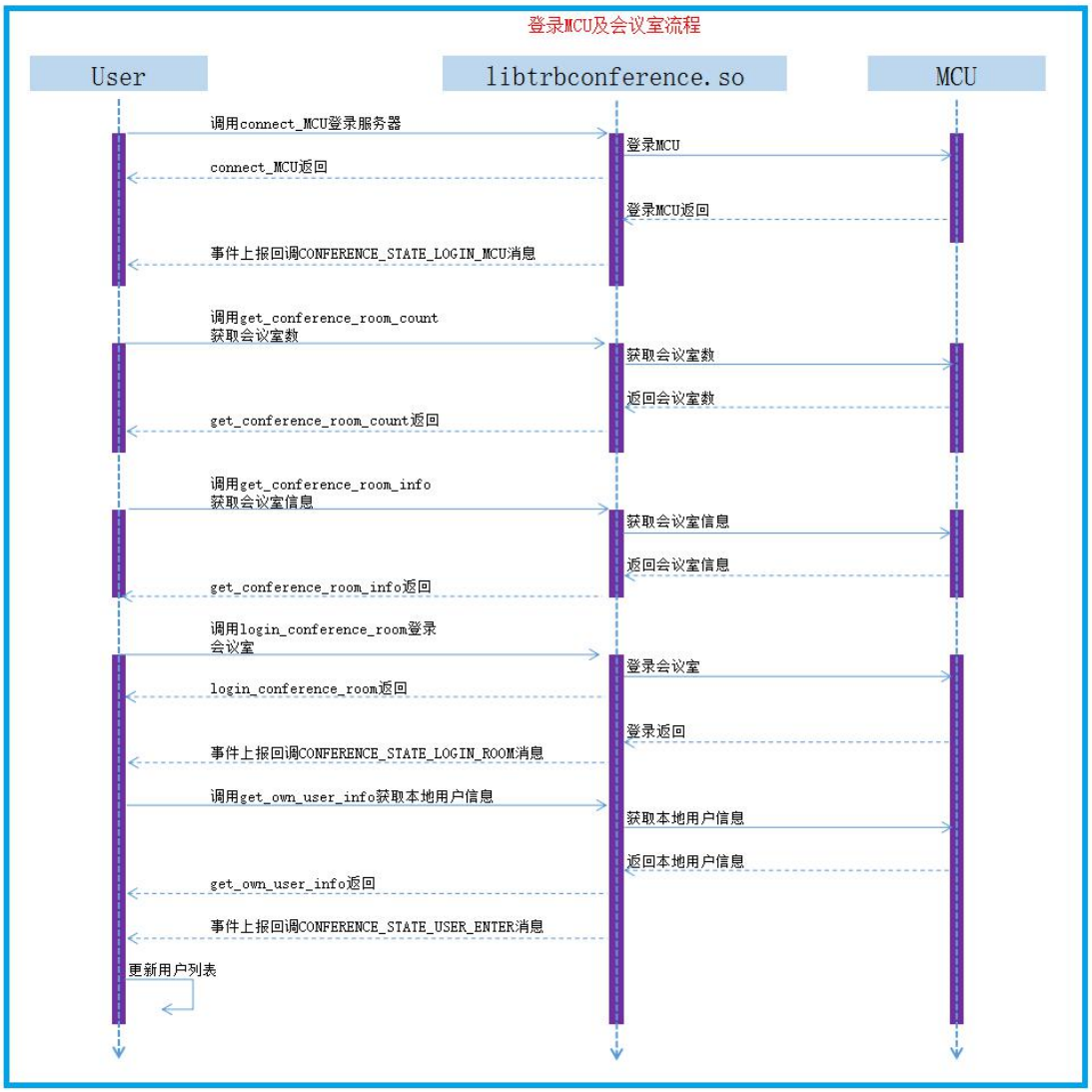
0x1001d	retCode_RoomOverdue	房间已过期
0x1001e	retCode_RoomClosed	房间已关闭
0x1001f	retCode_ModeratorRefusesJoin	主持人拒绝用户进入
0x10020	retCode_UuserAreBlackListed	用户被列入黑名单
0x10021	retCode_ServerCascadeFailure	服务器级联失败
0x10022	retCode_ServerNotBoot	服务器未启动
0x10023	retCode_ServerStoped	服务器已停止
0x10024	retCode_LackOfBalance	余额不足
0x10025	retCode_ServiceDiscontinuation	服务已停用
0x10026	retCode_ChairPasswordErr	主席密码错误
0x10027	retCode_InteractiveClassroomUnexist	互动课堂不存在
0x10028	retCode_AccessDenied	拒绝访问
0x10029	retCode_KickOut	踢出房间
0x1002a	retCode_UserClose	用户异常
0x1002b	retCode_StreamTypeMismatch	流类型不匹配
0x1002c	retCode_ExceedMaxPreview	超出最大预览数
0x1002d	retCode_OutOfResources	资源不足
0x1002e	retCode_NetworkException	网络异常
0x1002f	retCode_RegisterIdle	空闲/未注册
0x10030	retCode_Registering	注册中
0x10031	retCode_Registered	已注册
0x10032	retCode_DeRegistered	已注销
0x10033	retCode_RegisterFailed	注册失败
0x10034	retCode_ErrorNetwork	网络错误
0x10035	retCode_ErrorAuthFailed	认证失败
0x10036	retCode_UserBusy	用户忙
0x10037	retCode_LongTimeNoReSponse	长时间无应答
0x10038	retCode_LoginFailed	登录失败

五、调用流程描述：

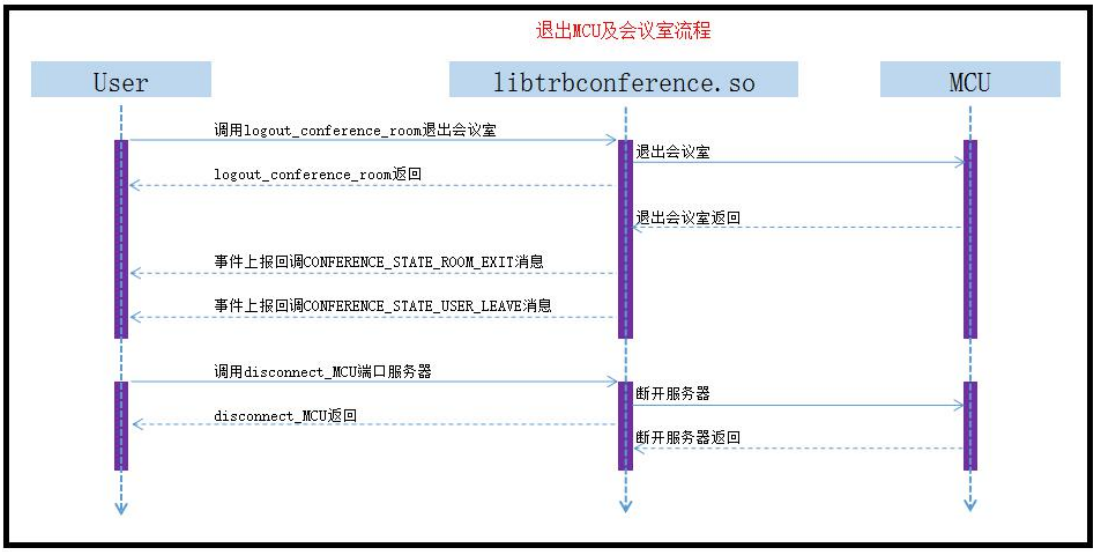
1、初始化流程：

1）定义回调函数；
2）创建 conference_event_handler_t 结构体，并关联回调函数；
3）调用 init 接口完成初始化。

2、登录 MCU 及会议室流程：



3、退出 MCU 及会议室流程：



4、其他控制流程（如申请发言、广播成员视频、查看成员视频等）：