楼宇对讲库接口说明

**v1.0**

**创建日期：2016-01-15**

**最后修改：2016-04-26**

**青岛东软载波科技股份有限公司**

目录

[楼宇对讲库接口说明 1](#_Toc449444826)

[1. 简介 4](#_Toc449444827)

[1.1. 楼宇对讲库 4](#_Toc449444828)

[1.2. UI(Android、QT) 4](#_Toc449444829)

[2. 楼宇对讲库 4](#_Toc449444830)

[2.1. 模块初始化顺序 4](#_Toc449444831)

[2.2. log系统 4](#_Toc449444832)

[2.3. 致命错误 5](#_Toc449444833)

[2.4. 楼宇对讲库初始化 5](#_Toc449444834)

[2.5. 销毁楼宇对讲库的服务 5](#_Toc449444835)

[2.6. 设备信息修改 6](#_Toc449444836)

[3. 对讲通信 6](#_Toc449444837)

[3.1. 初始化对讲服务 6](#_Toc449444838)

[3.2. 收到呼叫消息 6](#_Toc449444839)

[3.3. 收到设备找到消息 6](#_Toc449444840)

[3.4. 视频播放地址URL 7](#_Toc449444841)

[3.5. 向ui推送视频数据 7](#_Toc449444842)

[3.6. 呼叫过程中对方挂断 7](#_Toc449444843)

[3.7. ui应答呼叫请求 7](#_Toc449444844)

[3.8. 收到对讲应答 7](#_Toc449444845)

[3.9. 对讲确认 8](#_Toc449444846)

[3.10. 向ui推送音频数据 8](#_Toc449444847)

[3.11. ui发送音频数据 8](#_Toc449444848)

[3.12. ui挂断呼叫 8](#_Toc449444849)

[3.13. ui挂断对讲 8](#_Toc449444850)

[3.14. 对讲过程中对方挂断 9](#_Toc449444851)

[3.15. 主动呼叫用户 9](#_Toc449444852)

[3.16. 用户回复忙 9](#_Toc449444853)

[3.17. 呼叫超时 9](#_Toc449444854)

[3.18. 对讲任务出错 9](#_Toc449444855)

[3.19. 呼叫中心管理机 10](#_Toc449444856)

[3.20. 开锁 10](#_Toc449444857)

[3.21. 开锁确认 10](#_Toc449444858)

[3.22. 监视门口机 10](#_Toc449444859)

[3.23. 门口机监视确认 10](#_Toc449444860)

[3.24. 主动挂断监视 11](#_Toc449444861)

[3.25. 被动挂断监视 11](#_Toc449444862)

[3.26. 监视回复忙 11](#_Toc449444863)

[3.27. 请求监视超时 11](#_Toc449444864)

[4. IMP 11](#_Toc449444865)

[4.1. 初始化IMP任务 12](#_Toc449444866)

[4.2. 销毁IMP任务 12](#_Toc449444867)

[4.3. 修改IMP地址 12](#_Toc449444868)

[4.4. IMP文本推送 12](#_Toc449444869)

[4.5. IMP广告推送 12](#_Toc449444870)

[4.6. IMP回复 12](#_Toc449444871)

[4.7. 收到用户报警消息（中心管理机） 13](#_Toc449444872)

[4.8. 发送对讲记录 13](#_Toc449444873)

[4.9. 呼梯 13](#_Toc449444874)

[4.10. 发送IMP报警信息 13](#_Toc449444875)

[4.11. 发送中心管理机报警信息 14](#_Toc449444876)

[5. 安防报警 14](#_Toc449444877)

[5.1. 初始化报警任务 14](#_Toc449444878)

[5.2. 销毁报警任务 14](#_Toc449444879)

[5.3. 修改防区信息 14](#_Toc449444880)

[5.4. 报警回调 14](#_Toc449444881)

[6. 附录 14](#_Toc449444882)

[6.1. 设备类型 14](#_Toc449444883)

[6.2. 设备信息 15](#_Toc449444884)

[6.3. IMP广告推送 15](#_Toc449444885)

[6.4. 回调函数类型枚举 15](#_Toc449444886)

[6.5. 防区开关 16](#_Toc449444887)

[6.6. 对讲服务错误码 16](#_Toc449444888)

[6.7. 致命错误错误码 16](#_Toc449444889)

[6.8. IMP事件类型 17](#_Toc449444890)

[6.9. IMP回复结果编码 17](#_Toc449444891)

# 简介

此文档详细说明了楼宇对讲库对外接口及使用方法。

楼宇对讲库使用C语言编写，并编译成.a静态库。由于Android版程序是用java编写，且只能使用.so动态库，因此在楼宇对讲库的基础上封装一层JNI，并编译为.so动态库。

以下为楼宇对讲库实现的服务和ui程序应实现的接口功能

## 楼宇对讲库

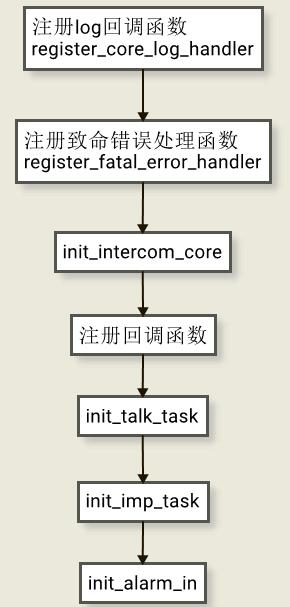
* 对讲通信，音视频传输
* 读取报警驱动，通知ui~~和imp~~报警消息
* 与IMP交互，将IMP的文本及广告推送通知ui

## UI(Android、QT)

* 对讲通信，音视频播放、录制
* 记录并读取对讲记录、安防报警记录
* 收到报警消息后声光报警
* 接收推送的文本消息和广告，向IMP发送对讲记录和安防报警消息

# 楼宇对讲库

## 模块初始化顺序



## log系统

楼宇对讲库的log系统独立于其他服务。如果ui注册了log回调函数，对讲库会使用注册的函数打印log，否则将log打印到标准输出。

ui应在启动其他服务前注册log回调函数。

注册函数：

int register\_core\_log\_handler(void (\*log\_handler)(int, char \*));

回调函数原型：core\_log(int priority, char \*msg)

回调函数参数：priority：log优先级；msg：log内容字符串

优先级枚举：

enum core\_log\_priority

{

CORE\_LOG\_PRI\_ERROR = 0,

CORE\_LOG\_PRI\_WARN,

CORE\_LOG\_PRI\_NOTICE,

CORE\_LOG\_PRI\_INFO,

CORE\_LOG\_PRI\_DEBUG,

CORE\_LOG\_PRI\_TRACE,

};

## 致命错误

对讲库出现致命错误时会回调致命错误的处理函数，致命错误包括加入组播失败、初始化IMP任务失败等，详细的错误码见附录。

ui需要在初始化对讲库前先注册致命错误处理函数。

注册函数：int register\_fatal\_error\_handler(void (\*error\_handler)(int));

回调函数原型：void fatal\_error\_handler(int error\_code)

回调函数参数：error\_code：错误码，详见附录

## 楼宇对讲库初始化

UI程序启动后，需要初始化楼宇对讲库

函数原型：int init\_intercom\_core(device\_info\_t \*device);

函数说明：参数device包含本设备的设备信息，详情参考“结构体定义”；返回值为0表示初始化成功，-1表示失败

## 销毁楼宇对讲库的服务

如果需要，ui可以销毁楼宇对讲库的服务，即各服务的线程和资源

函数原型：int destory\_intercom\_core(void);

函数说明：返回值为0表示初始化成功，-1表示失败

## 设备信息修改

函数原型：int modefy\_device\_info(device\_info\_t \*device);

函数说明：device\_info\_t设备信息结构体；返回值为0表示初始化成功，-1表示失败

# 对讲通信

## 初始化对讲服务

Android程序启动后，需要初始化对讲服务。

函数原型：int init\_talk\_task(void);

函数说明：初始化成功返回值为0，失败时返回值为-1

## 收到呼叫消息

对讲服务收到呼叫消息后，通过回调通知ui：

ui需要事先通过下面的函数注册回调函数

注册函数：

void register\_inactive\_call\_handler(void (\*inactive\_call\_handler)(device\_info\_t \*));

回调函数原型：void inactive\_call\_handler(device\_info\_t \*device\_info)

回调函数说明：参数device \_info包含发起呼叫方的设备类型和对应的设备信息，详情参考“结构体定义”

## 收到设备找到消息

主动呼叫时需要先查找设备，ui显示“正在查找设备”，设备找到后改为“正在呼叫”。找到设备后对讲服务以回调方式通知ui。

注册函数：

void register\_call\_answer\_found\_handler(void (\*call\_answer\_found\_handler)(device\_info\_t \*));

回调函数原型：void call\_answer\_found\_handler(device\_info\_t \*device\_info)

回调函数说明：参数device \_info包含对方的设备类型和对应的设备信息，详情参考“结构体定义”

## 视频播放地址URL

对讲服务收到视频帧后，如果是Android版室内机则建立Rtsp服务，并通过回调函数通知ui视频播放地址

注册函数：int register\_video\_url\_handler(void (\*url\_handler)(char \*));

回调函数原型：void video\_url\_handler(char \*url)

回调函数说明：url为“\0”结尾的字符串

## 向ui推送视频数据

对讲服务向ui推送视频数据，目前为92755方案专用。

对讲服务收到完整的一帧视频数据后，检查回调函数是否注册，如果已注册了回调函数，则将数据交给回调函数处理，不再创建rtsp服务。

注册函数：

void register\_play\_video\_frame\_handler(void (\*play\_video\_handler)(char \*, int));

函数原型：void play\_video\_frame\_handler(char \*data, int len)

函数说明：参数data为一帧完整的视频帧

## 呼叫过程中对方挂断

呼叫过程中（即正在振铃，还未接听），如果对方主动挂断，则对讲服务通过回调通知ui此消息

注册函数：

void register\_inactive\_call\_hang\_up\_handler(void (\*inactive\_call\_hang\_up\_handler)(void));

回调函数原型：void inactive\_call\_hang\_up\_handler(void)

## ui应答呼叫请求

ui应答呼叫请求时需主动调用对讲服务的应答呼叫函数

函数原型：int ui\_talk\_answer(void);

函数说明：返回值为0表示应答成功，-1表示应答失败

## 收到对讲应答

主动呼叫用户时，对方应答呼叫后，对讲服务通过回调通知ui开始对讲

注册函数原型：

void register\_talk\_answer\_handler(void (\*talk\_answer\_handler)(void));

回调函数原型：void talk\_answer\_handler(void)

## 对讲确认

对讲服务收到对方确认对讲的报文后，通过回调通知ui开始对讲

注册函数：

void register\_talk\_confim\_handler(void (\*talk\_confim\_handler)(void));

回调函数原型：void talk\_confim\_handler(void)

## 向ui推送音频数据

对讲服务收到音频数据后，通过回调函数通知ui处理音频数据

注册函数：

void register\_audio\_data\_handler(void (\*audio\_handler)(int ,char \*));

回调函数原型：audio\_data\_handler(int len, char \*data)

回调函数说明：len：音频数据长度，\*data：音频数据流

## ui发送音频数据

ui录制一段音频（和视频帧对应长度）后，调用对讲服务的函数发送数据

函数原型：void push\_audio\_data(int len, char \*data);

函数说明：len：发送音频数据的长度，\*data：音频数据流

## ui挂断呼叫

ui挂断呼叫时，需要调用对讲服务的挂断呼叫函数

函数原型：int call\_hang\_up(void)

函数说明：返回值为0表示成功，其他为失败

## ui挂断对讲

ui挂断对讲时，需要调用对讲服务的挂断对讲函数

函数原型：int talk\_hang\_up(void)

函数说明：返回值为0表示成功，其他为失败

## 对讲过程中对方挂断

对讲过程中如果对方主动挂断，则对讲服务通过回调通知ui此消息

注册函数：

void register\_inactive\_talk\_hang\_up\_handler(void (\*inactive\_talk\_hang\_up\_handler)(void));

回调函数原型：void inactive\_talk\_hang\_up\_handler(void)

## 主动呼叫用户

函数原型：int active\_call\_user(device\_info\_t \*device)

函数说明：参数device包含被呼叫用户的设备信息，详情参考“结构体定义”

## 用户回复忙

呼叫用户时，如果被叫设备回复忙，则对讲服务回调呼叫用户忙的函数。

注册函数：

void register\_call\_busy\_handler(void (\*call\_busy\_handler)(void));

函数原型：void call\_busy\_handler(void)

## ~~呼叫超时~~

~~呼叫超时时（包括呼叫用户和中心管理机），对讲服务会回调呼叫超时处理函数。~~

~~注册函数：~~

~~void register\_active\_call\_timeout\_handler(void (\*call\_timeout\_handler)(device\_info\_t \*));~~

~~函数原型：void active\_call\_timeout\_handler(device\_info\_t \*device)~~

~~函数说明：参数device为被呼叫设备的设备信息~~

## 对讲任务出错

对讲任务出错主要包含各种超时错误，错误类型详见错误码

注册函数：

void register\_talk\_task\_error\_handler(void (\*error\_handler)(int, device\_info\_t \*));

函数原型：void talk\_task\_error\_handler(int error\_code, device\_info\_t \*peer)

函数说明：参数error\_code为错误码，详见附录；参数peer为对方设备的设备信息

## 呼叫中心管理机

函数原型：int call\_center\_manager(struct in\_addr \*addr);

函数说明：addr为中心管理机的ip地址

## 开锁

呼叫和对讲过程中室内机可以向门口机发送开锁命令

函数原型：int unlock\_door(void);

函数说明：返回值为0表示发送开锁报文成功，-1表示失败

## 开锁确认

发送开锁报文后，门口机开锁成功会回复确认报文，此时对讲库调用开锁确认的回调函数

注册函数：

void register\_unlock\_door\_confirm\_handler(void (\*unlock\_confirm\_hanler)(int));

回调函数原型：void unlock\_door\_confirm\_handler(int result)

回调函数说明：参数result:1-开锁成功; 0-超时

## 监视门口机

函数原型：int ui\_req\_monitor(device\_info\_t \*device);

函数说明：参数device\_info\_t为设备信息结构体，参加“结构体定义”

## 门口机监视确认

门口机接收到监视请求后，回复监视确认。

注册函数：

void register\_monitor\_confirm\_handler(void (\*monitor\_confirm\_handler)(device\_info\_t \*));

函数原型：void monitor\_confirm\_handler(device\_info\_t \*device);

函数说明：参数device\_info\_t为设备信息结构体，参加“结构体定义”

## 主动挂断监视

监视过程中，用户主动挂断监视。

函数原型：int active\_hang\_up\_monitor(device\_info\_t \*device);

函数说明：参数device\_info\_t为设备信息结构体，参加“结构体定义”

## 被动挂断监视

如果在监视过程中，门口机或围墙机呼叫用户，则门口机或围墙机请求挂断监视。

注册函数：

void register\_inactive\_hang\_up\_monitor\_handler(void (\*hang\_up\_handler)(device\_info\_t \*));

函数原型：void inactive\_hang\_up\_monitor\_handler(device\_info\_t \*device);

函数说明：参数device\_info\_t为设备信息结构体，参加“结构体定义”

## 监视回复忙

用户监视门口机或围墙机，如果将被监视的设备正在通话中，则恢复忙。

注册函数：

void register\_monitor\_response\_busy\_handler(void (\*monitor\_busy\_handler)(device\_info\_t \*));

函数原型：void monitor\_response\_busy\_handler(device\_info\_t \*device);

函数说明：参数device\_info\_t为设备信息结构体，参加“结构体定义”

## 请求监视超时

请求监视门口机/围墙机超时后，对讲服务会回调超时处理函数。

注册函数：

void register\_req\_monitor\_timeout\_handler(void (\*req\_monitor\_timeout\_handler)(device\_info\_t \*));

函数原型：void req\_monitor\_timeout\_handler(device\_info\_t \*device)

参数说明：参数device为被监视设备的设备信息

# IMP

## 初始化IMP任务

函数原型：int init\_imp\_task(struct in\_addr \*imp\_addr);

函数说明：参数imp\_addr为IMP地址；返回值为0表示初始化成功，-1表示初始化失败

## 销毁IMP任务

函数原型：int destory\_imp\_task(void);

函数说明：返回值为0表示销毁成功，-1表示销毁失败

## 修改IMP地址

函数原型：int modify\_imp\_addr(struct in\_addr \*imp\_addr);

函数说明：返回值为0表示修改成功，-1表示修改失败

## IMP文本推送

注册函数：

int register\_text\_push\_handler(void (\*text\_push\_handler)(char \*));

回调函数原型：void imp\_text\_push\_handler(char \*text)

回调函数说明：参数\*text为推送文明字符串，带“\0”结束符

## IMP广告推送

注册函数：

int register\_ad\_push\_handler(void (\*ad\_push\_handler)(imp\_ad\_push\_t \*));

回调函数原型：void imp\_ad\_push\_handler(imp\_ad\_push\_t \*ad)

回调函数说明：参数imp\_ad\_push\_t为广告推送结构体，相见“结构体定义”

## IMP回复

ui主动发送的IMP报文（比如呼梯、安防报警），如果对讲库都到了IMP的回复报文，会通过此接口通知ui。

注册函数：

int register\_imp\_response\_handler(void (\*response\_handler)(int, int));

回调函数原型：void imp\_response\_handler(int event\_type, int result)

回调函数说明：event\_type：事件类型；result：IMP回复结果。

时间类型枚举：

**enum** imp\_event\_type\_enum

{

*IMP\_ET\_CALL\_RECORD* = 1,

*IMP\_ET\_ALARM\_IN*,

*IMP\_ET\_ELEVATOR\_GUEST*,

*IMP\_ET\_ELEVATOR\_CALL*,

};

## 收到用户报警消息（中心管理机）

对讲库收到用户报警消息后，通过此回调函数通知ui，如果设备不是中心管理机，ui就不应该注册此函数。

注册函数：

int register\_user\_alarm\_trigger\_handler(void (\*alarm\_handler)(device\_info\_t \*,int, int));

回调函数原型：

void user\_alarm\_trigger\_handler(device\_info\_t \*device, int type, int data)

回调函数说明：device：报警设备的设备信息；type：报警类型；data：报警数据（0撤警，1报警）

## 发送对讲记录

主动发起对讲的设备应向IMP发送对讲记录，UI在对讲完成后调用本接口发送对讲记录

函数原型：int send\_imp\_call\_record(device\_info\_t \*peer\_info, int is\_answer, int is\_unlock);

函数说明：返回值为0表示发送成功，其他表失败；参数peer\_info为对方设备的设备信息；参数is\_answer为1表示已接听，0表示未接听；参数is\_unlock为1表示已开锁，0表示未开锁

## 呼梯

函数原型：int send\_imp\_call\_elevator(int floor, int room)

函数说明：floor：呼梯楼层；root：呼梯房间

## 发送IMP报警信息

ui收到报警后，要进行声光报警，记录本地报警记录，并使用此接口向imp发送报警消息。

函数原型：int send\_imp\_alarm\_in\_record(int type, int data)

函数说明：type：0~n 防区类型；data：0撤警，1报警

## 发送中心管理机报警信息

ui收到报警后，使用此接口向中心管理机发送报警信息。

函数原型：int send\_center\_manager\_alarm\_record(struct in\_addr \*addr, device\_info\_t \*peer\_info, int type, int data)

函数说明：addr:中心管理机地址；peer\_info:中心管理机设备信息；type:0~n 防区类型；data:0撤警，1报警

# 安防报警

## 初始化报警任务

函数原型：int init\_alarm\_in(defences\_switch\_t \*defence)

函数说明：参数defence描述防区总开关和各防区开关

## 销毁报警任务

函数原型：int destory\_alarm\_in(void);

## 修改防区信息

函数原型：int modify\_alarm\_in\_defences\_area(defences\_switch\_t \*defence)

函数说明：参数defence描述防区总开关和各防区开关

## 报警回调

注册函数：

int register\_alarm\_handler(void (\*alarm\_handler)(int));

回调函数原型：void alarm\_in\_handler(int channel)

回调函数说明：参数channel为报警的通道

# 附录

## 设备类型

**typedef** **enum** device\_type\_t

{

*DT\_Room\_Machine* = 0,

*DT\_Unit\_Door\_Machine*,

*DT\_Wall\_Machine*,

*DT\_Secondary\_Confirmation\_Machine*,

*DT\_Center\_Machine*,

}device\_type\_t;

## 设备信息

**typedef** **struct** device\_info\_s device\_info\_t;

**struct** device\_info\_s

{/\*结构体成员的排列顺序和网络报文相关，因此不可随便颠倒顺序\*/

**int** building\_no;

**char** unit\_no;

**char** layer\_no;

**char** room\_no;

**unsigned** **int** dev\_no;

**unsigned** **int** device\_type;

};

## IMP广告推送

**typedef** **struct** imp\_ad\_push\_t

{

**int** enable;

**int** play\_time;

**char** url[IMP\_AD\_PUSH\_URL\_MAX\_LEN];

}imp\_ad\_push\_t, \*p\_imp\_ad\_push\_t;

## 回调函数类型枚举

**typedef** **enum** callback\_handler\_type\_t

{

*CALLBACK\_ALARM\_IN* = 100,

*CALLBACK\_IMP\_TEXT\_PUSH* = 200,

*CALLBACK\_IMP\_AD\_PUSH*,

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* talk \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*CALLBACK\_INACTIVE\_CALL* = 300,

*CALLBACK\_VIDEO\_URL*,

*CALLBACK\_PLAY\_VIDEO\_FRAME*,

*CALLBACK\_AUDIO\_DATA*,

*CALLBACK\_INACTIVE\_CALL\_HANG\_UP*,

*CALLBACK\_INACTIVE\_TALK\_HANG\_UP*,

*CALLBACK\_TALK\_CONFIM*,

*CALLBACK\_CALL\_RESPONSE\_BUSY*,

*CALLBACK\_UNLOCK\_CONFIRM*,

*CALLBACK\_CALL\_ANSWER\_FOUND*,

*CALLBACK\_TALK\_ANSWER*,

*CALLBACK\_TALK\_TASK\_ERROR*,

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* monitor \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*CALLBACK\_MONITOR\_CONFIRM*,

*CALLBACK\_INACTIVE\_HANG\_UP\_MONITOR*,

*CALLBACK\_MONITOR\_RESPONSE\_BUSY*,

*CALLBACK\_REQ\_MONITOR\_TIMEOUT*,

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* fatal error \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*CALLBACK\_FATAL\_ERROR* = 1000,

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* log \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

*CALLBACK\_CORE\_LOG* = 1100,

}callback\_handler\_type\_t;

## 防区开关

**#define** MAX\_DEFENSE\_AREA\_NUM 8

**typedef** **struct** defences\_switch\_s

{

**int** onoff; //总开关

**char** area[MAX\_DEFENSE\_AREA\_NUM]; //1:防区开，0:防区关

}defences\_switch\_t;

## 对讲服务错误码

**enum** talk\_task\_error\_code\_e

{//按照袁堂亮的要求，呼叫用户超时和接听超时都使用talk\_task\_error回调函数

*TTE\_CALL\_USER\_TIMEOUT* = 1, //包括呼叫用户，呼叫中心管理机

*TTE\_TALK\_ANSWER\_TIMEOUT*,

};

## 致命错误错误码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* fatal error

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**enum**

{

*FE\_NET\_JOIN\_GROUP* = 1,

*FE\_INIT\_IMP\_FAILED* = 101,

*FE\_INIT\_ALARM\_TASK\_FAILED* = 201,

*FE\_OPEN\_ALARM\_DRIVER\_ERROR* = 202,

*FE\_OTHER\_ERROR*,

};

## IMP事件类型

**enum** imp\_event\_type\_enum

{

*IMP\_ET\_CALL\_RECORD* = 1,

*IMP\_ET\_ALARM\_IN*,

*IMP\_ET\_ELEVATOR\_GUEST*,

*IMP\_ET\_ELEVATOR\_CALL*,

};

## IMP回复结果编码

**enum** imp\_result\_enum

{

*IMP\_RESULT\_TIMEOUT* = -1,

*IMP\_RESULT\_OTHRE\_ERROR* = -2,

*IMP\_RESULT\_SUCCESS* = 200,

*IMP\_RESULT\_FAILED* = 480,

};