CH人脸识别SDK说明书

(For winXP/2003/Vista/win7)

版本**1.3 beta** **2012-12-4**

申　　明：本文档介绍如何通过SDK操作STONELOCK人脸识别机，该文档为内部机密资料，任何获得该文档的个人和公司不得泄露该文档。

版本说明**: ver1.3**

1. 通过组播方式自动搜索所有加入该组播地址的人脸机IP，支持指定IP访问人脸机，所有人脸机默认组播地址 224.0.1.100，可通过API修改。
2. 获得/设置人脸机工作属性DEV_WORKATT，获得人脸机统计信息DEV_STATIS等
3. 批量上传，下载，修改用户资料
4. 批量下载各种类型记录(识别记录，报警记录，变更记录，操作日志)
5. 支持时间组，权限，组合开门。
6. 支持实时记录，远程用户加载，用户实时同步，远程用户采集，远程视频查看
7. 远程开/关门，常开/常闭，远程重启，格式化，固件网络升级
8. 支持N+1，AB互锁，门磁状态实时反馈等专业门禁功能
9. 兼容本地模式和中转模式，本地模式，只支持一个客户端连接设备，中转模式下，通过中转服务器，可以支持客户端与设备N-N操作。

重要说明:

1. 人脸机设计为单一网络连接，也即同时只能有一个管理客户端能与人脸机通讯。 需要多客户端时，可选择中转模式，通过中转服务器进行N-N型的操作。
2. 所有用户以用户ID(编号)为唯一关键字，不可重复。
3. 用户ID，姓名，部门等用DEV_CID表示的字符串长度不能超过23个BYTES，字符集采用多字节字符集。
4. 关于变更注册照，每一次用户采集无论是新增还是重采都会用用户最新的彩照作为他的注册照，除非注册照为管理端导入。该策略主要解决识别历史记录与注册照的对应问题。
5. 所有记录的流水号都是全局唯一的，组成: SN+TYPE+时间+随机数
6. 每个命令操作，用户可指定唯一标识号(操作序列)，用于操作结果返回时与命令精确匹配，序列号0为内部保留，外部不能使用。
7. 大量设备在线(>10台)时，大批量上传下载数据，建议采用任务队列形式，限制同时不超过5台设备进行数据上传下载。

关于记录和事件来源485地址解析:

1. 485地址高4位[4-7]表示设备索引号1-15，低4位[0-3]表示支持16种不同型的设备

```
// #define DEV_TYPE_FACE1      0x0        //人脸前端1
// #define DEV_TYPE_FACE2      0x1        //人脸前端2
// #define DEV_TYPE_CARD1      0x2        //刷卡器1
// #define DEV_TYPE_CARD2      0x3        //刷卡器2
// #define DEV_TYPE_IN         0x4        //辅助输入
// #define DEV_TYPE_OUT        0x5        //辅助输出
// #define DEV_TYPE_INWL       0x6        //无线输入
// #define DEV_TYPE_DOOR       0xF        //门点本身
//  例如: 0x11  表示1号门人脸识别前端1，INDEX[4-7] == 1   TYPE[0-3] == 1
//          0x10  表示1号门人脸识别前端0，INDEX[4-7] == 1   TYPE[0-3] == 0
//          0x12  表示1号门刷卡器0          INDEX[4-7] == 1   TYPE[0-3] == 2
//          0x13  表示1号门刷卡器1          INDEX[4-7] == 1   TYPE[0-3] == 3
//          0x1F  表示门点本身              INDEX[4-7] == 1   TYPE[0-3] == F
//          0x14  表示辅助输入1             INDEX[4-7] == 1   TYPE[0-3] == 4
//          0x25  表示辅助输出2             INDEX[4-7] == 2   TYPE[0-3] == 5
//2. 所有设备的序号从1开始。此协议可支持15张门，16中不同类型的门点输入设备，15个辅助输
```

入和15个辅助输出
//3. 特殊意义的地址: [0-7] == 0x00 表示非法地址，[0-7] ＝＝ 0x01表示后端板本身, [0-7] ＝＝ 0x02表示GPRS模块
//7_____3_____0
//|_INDEX_|__type__|

数据结构详解:

1. 宏定义

宏名	值	解释
DEV_REGION_ADDR	"224.0.1.100"	默认组播地址(可利用该地址划分区域)，注意需要路由器支持组播功能
DEV_ID_LEN	24	ID字符串长度(<23bytes)
DEV_TIMEGROUP_NUMS	8	一个权限最多支持的时间组个数
DEV_USER_COMBINS	6	一个开门组合最多支持6个用户
DL_DEV_PIC	0x00000001	下载注册照片 需和DL_DEV_USER_TEXT组合使用
DL_DEV_USER_FEAT	0x00000002	下载人脸特征 需和DL_DEV_USER_TEXT组合使用
DL_DEV_USER_TEXT	0x00000004	下载文字信息，可与DL_DEV_PIC|DL_DEV_USER_FEAT组合使用
DEV_CHECK_TIME	0x01	时间组中需要检测时间段
DEV_CHECK_WEEK	0x02	时间组中需要检测星期
DEV_CHECK_DAY	0x04	时间组中需要检测日期
DEV_CHECK_MONTH	0x08	时间组中需要检测月份
DEV_CHECK_YEAR	0x10	时间组中需要检测年份
DEV_WEEK_1	0x01	检测星期时，星期一有效
DEV_WEEK_2	0x02	检测星期时，星期二有效
DEV_WEEK_3	0x04	检测星期时，星期三有效
DEV_WEEK_4	0x08	检测星期时，星期四有效
DEV_WEEK_5	0x10	检测星期时，星期五有效
DEV_WEEK_6	0x20	检测星期时，星期六有效
DEV_WEEK_7	0x40	检测星期时，星期天有效
DEV_RECORD_SAVEFAIL	0x00000001	保存识别失败记录
DEV_SUPER_PASSWORD	0x00000002	超级密码开门有效
DEV_HDBEEP_OPEN	0x00000004	布防/撤防
DEV_REALTIME_RECORD	0x00000010	实时动态记录显示
DEV_REALTIME_USERLOAD	0x00000020	输入工号或者卡号识别时，如果无法在当前人脸机中获得人脸特征，将尝试从管理端远程加载
DEV_REALTIME_USERSEND	0x00000040	采集用户实时发送
DEV_DOORMANGET_OPEN	0x00000080	使能门磁检测报警
DEV_DOORFORCE_OPEN	0x00000100	使能胁迫开门功能
DEV_REMOTE_CAP_SAVE	0x00000200	通过管理端远程采集的用户是否保存到人脸机本地
DEV_GPRS_OPEN	0x00000400	暂未使用
DEV_UPDATE_USERSEND	0x00000800	特征更新时，是否实时发送用户特征

DEV_AB_LOCK	0x00002000	使能AB互锁
DEV_DOOR1_NOPEN	0x00004000	门一N+1
DEV_DOOR2_NOPEN	0x00008000	门二N+1
DEV_VERIFY_USERID	0x01	需要输入编号ID
DEV_VERIFY_CARD	0x02	需要刷卡
DEV_VERIFY_FACE_11	0x04	1:1 人脸识别，需要和DEV_VERIFY_USERID DEV_VERIFY_CARD组合使用
DEV_VERIFY_FACE_MIX	0x08	混合人脸识别，可与DEV_VERIFY_USERID DEV_VERIFY_CARD组合使用
DEV_VERIFY_FACE_1N	0x10	1:N 人脸识别，可与DEV_VERIFY_USERID DEV_VERIFY_CARD组合使用
DEV_USER_CARD_INVALID	0x01	用户卡已挂失
DEV_USER_DLine_INVALID	0x02	失效时间有效
DEV_USER_BLACK	0x04	用户为黑名单用户
DEV_USER_MIX1N	0x08	混合模式下，该用户进行1:N模式验证
DEV_USER_FEAT_BASE64	0x40	特征经过base64编码，与B/S结合有关
DEV_USER_IMAGE_BASE64	0x80	注册照经过base64编码，与B/S结合有关

2. 枚举类型

2.1 网络连接状态

```
typedef enum
{
        DEV_CONNECT_CUT = 0x00,        //断开连接
        DEV_CONNECT_SUCCESS,           //建立连接
        DEV_CONNECT_FAILUE ,           //连接失败
        DEV_CONNECT_NONE               //未建立连接
}DEV_CONNECT_STATUS;
```

2.2 设备数据操作标识

```
typedef enum
{
        DEV_OPER_UPLOAD = 0x01,          //上传
        DEV_OPER_DOWNLOAD = 0x02,        //下载
        DEV_OPER_DELETE     = 0x04       //删除
}DEV_OPER_FLAG;
```

2.3 设备数据具体操作类型

```
typedef enum{
        DEV_AUTH_OPER = 0x01,          //设备验证操作
        DEV_AUTH_SET_OPER,             //设备验证用户设置操作
        DEV_REGION_OPER ,              //设备区域操作
        DEV_FORMAT_OPER,               //设备格式化操作
        DEV_UPDATE_OPER,               //设备程序更新操作
```

```
        DEV_SYS_TIME_OPER ,          //设备系统时间操作
        DEV_BASEINFO_OPER,           //基本信息操作
        DEV_STATISINFO_OPER,         //统计信息操作
        DEV_WORKATT_OPER,            //工作属性操作
        DEV_USER_OPER ,              //用户操作
        DEV_USER_RT_OPER,            //用户实时加载操作
        DEV_USER_RANGE_OPER,         //用户下载数据条数回馈操作
        DEV_RECORD_OPER    ,         //记录操作
        DEV_RECORD_RANGE_OPER,       //记录下载数据条数回馈操作
        DEV_ALARM_OPER,
        DEV_ALARM_RANGE_OPER,
        DEV_LOG_OPER,                //日志操作
        DEV_LOG_RANGE_OPER,          //日志下载数据条数回馈操作
        DEV_CHANGEIMAGE_OPER,        //变更注册照操作
        DEV_CIMAGE_RANGE_OPER,       //变更注册照下载数据条数回馈操作
        DEV_TIME_GROUP_OPER,         //时间组操作
        DEV_RIGHT_OPER,              //权限操作
        DEV_USERGROUP_OPER,          //用户组操作
        DEV_DOOR_STATE_OPER,         //开门状态操作
        DEV_REMOTE_OPEN_DOOR,        //远程开门操作
        DEV_VIDEO_TRANSFER,          //视频传输
        DEV_USER_EX_OPER,            //用户批量操作
        DEV_RECORD_EX_OPER,          //记录批量操作
        DEV_LOG_EX_OPER,             //日志批量操作
        DEV_CIMAGE_EX_OPER,          //变更注册照批量操作
        DEV_REBOOT_OPER,             //设备重启
        DEV_USER_REMOTE_CAP,         //用户远程采集
        DEV_NOPEN_OPER,              //N+1开门
        DEV_IOCTRL_OPER,             //IOCTRL控制信息
        CLI_AUTH_OPER,               //client像中转服务器验证操作
}DEV_OPER_TYPE;
```

2.4 设备操作结果

```
typedef enum
{
        OPER_SUCCESS      =0x00,          //成功
        OPER_ERR_BUSY     =0x01,          //设备忙
        OPER_ERR_LIMIT    =0x02,          //已达上限
        OPER_ERR_NOFIND   =0x03,          //没有找到对应数据
        OPER_ERR_SAVEFAIL =0x04,          //数据保存失败
        OPER_ERR_SETFAIL  =0x05,          //设置失败
        OPER_ERR_FROMAT   =0x07,          //格式化失败
        OPER_ERR_PARAM    =0x08,          //参数错误
        OPER_ERR_DISABLE  =0x09,          //要求执行的功能没有使能
        OPER_ERR_EXECUT   =0x0A,          //失败
        OPER_ERR_SUPPORT  =0x10,          //不支持的命令
```

```
        OPER_ERR_INPUTDATA =0x11,           //网络端传输的数据有异常
        OPER_BATCH_DL_COMPLETE = 0x1F     //批量下载完成
}DEV_OPER_RESULT;
```

2.5 时间组类型

```
typedef enum
{
        DEV_NORMAL    = 0x01,   //正常时间
        DEV_HOLIDY     = 0x02,   //节假日
        DEV_ANY_TIME = 0x03    //任意时间
}DEV_TIMEGROUP_TYPE;
```

2.6 用户类型

```
typedef enum
{
        DEV_USER_NOMAL = 0,          //普通
        DEV_USER_ADMIN = 1,          //管理员
        DEV_USER_SUPERADMIN = 2   //超级管理员
}DEV_USER_TYPE;
```

2.7 用户操作设备默认权限

```
typedef enum
{
        DEV_DEFAULT_NO_RIGHT       = -2,       //无权限
        DEV_DEFAULT_SINGLE_OPEN = -1         //单一开门权限
}DEV_DEFAULT_RIGHT;
```

2.7 记录类型

```
//0x01 - 0x20:  日常记录
//0x20 - 0xFF:  报警记录
typedef enum
{
        DEV_VERIFY_SUCC = 0x01,                          //验证成功
        DEV_VERIFY_FAIL = 0x02,                          //验证失败
        DEV_ADMIN_SUCC = 0x03,                           //管理员验证成功
        DEV_EMER_OPEN = 0x04,                            //紧急开门
        DEV_RIGHT_OPEN = 0x05,                           //权限开门
        DEV_GROUP_OPEN = 0x06,                           //组合开门
        DEV_BUTTON_OPEN = 0x07,                          //按钮开门
        DEV_ALARM_HD_MANGET_TIMEOUT = 0x20, //门磁超时
        DEV_ALARM_HD_MANGET_ILLOPEN = 0x21,   //门磁非法开门
        DEV_ALARM_HD_OFFLINE = 0x22,                     //前端掉线报警
        DEV_ALARM_HD_BREAK = 0x30,                       //防拆报警
        DEV_ALARM_HD_SHOCK =    0x31,                    //震动报警
        DEV_ALARM_HD_FPOWR = 0x36,                       //前端供电异常报警
        DEV_ALARM_HD_UPS_ON = 0x37,                      //UPS备用电池开启
```

```
        DEV_ALARM_HD_UPS_OFF = 0x38,              //UPS备用电池关闭
        DEV_ALARM_HD_ASSIST = 0x40,               //辅助输入触发
        DEV_ALARM_SF_BACKLIST = 0xF0,             //黑名单验证报警
        DEV_ALARM_SF_ILLCARD = 0xF1,              //无效卡(挂失)
        DEV_ALARM_SF_ILLTIME = 0xF2,              //非法时间识别
        DEV_ALARM_SF_DEADLINE = 0xF3,             //失效时间
        DEV_ALARM_SF_DANGER_OPEN = 0xF4,          //胁迫开门
        DEV_ALARM_SF_SUPER_OPEN = 0xF5            //超级密码开门
}DEV_REC_TYPE;
```

2.8 记录来源

```
typedef enum
{
        DEV_DOOR_SOURCE = 0x01,           //门相关来源
        DEV_SIGNAL_SOURCE,                //信号输入输出相关来源
        DEV_NO_NUM_DEV_SOURCE             //无编号设备来源
}DEV_REC_SOURCE_TYPE;
```

2.9 操作日志来源

```
typedef enum
{
        DEV_LOG_ADDUSER = 0x01,           /增加用户
        DEV_LOG_DELUSER = 0x02,           //删除用户
        DEV_LOG_REREGIST = 0x03,          //重新采集
        DEV_LOG_CAHNGETYPE = 0x04,        //改变用户类型
        DEV_UDISK_ULUSER = 0x05,          //U盘上传用户
        DEV_UDISK_DLUSER = 0x06,          //U盘下载用户
        DEV_UDISK_DLRECORD = 0x07,        //U盘下载记录
        DEV_UDISK_UPDATEAPP = 0x08        //U盘更新程序
}DEV_LOG_TYPE;
```

2.10门点状态

```
typedef enum
{
        DEV_DOOR_NOMAL = 0x00000000,      //正常状态
        DEV_DOOR_OPEN = 0x00000100,    /     /开状态
        DEV_DOOR_CLOSE = 0x00000200       //关状态
}DEV_DOOR_STATE;
```

3. 结构体类型

3.1 ID标识结构体

```
typedef struct
{
    char m_ID[ DEV_ID_LEN ];
}DEV_CID;
```

3.2 日期结构体

```
typedef struct
{
    int     m_Year;
    char    m_Month;
    char    m_Day;
}DEV_DATE;
```

3.3 时间结构体

```
typedef struct
{
    int     m_Msec;             //毫秒
    char    m_Hour;
    char    m_Minute;
    char    m_Second;
}DEV_TIME; //时间
```

3.4 日期和时间结构体

```
typedef struct
{
    DEV_DATE m_Date;
    DEV_TIME m_Time;
}DEV_DATETIME;
```

3.4 设备基本信息

```
typedef struct
{
    int    m_DevType;           //设备类型
    int    m_LimitUser;         //总使用人数上限
    int    m_Limit1NUser;       //1N用户上限
    DEV_CID m_SN;               //设备编号
    DEV_CID m_Ver;              //DSP软件版本
    DEV_CID m_Space;            //磁盘容量信息
}DEV_BASEINFO;
```

3.5 设备统计信息

```
typedef struct
{
    int    m_TotalUsers;            //当前设备的总用户数
    int    m_NoFeatUser;            //没有采集人脸特征的用户数
```

```
    int    m_TotalDays;              //识别记录保存总天数
    int    m_TotalRecords;           //总记录数
    int    m_TotalAlarm;             //总报警记录数
    int    m_TotalDspLog;            //总操作日志数目
    int    m_TotalChangeImage;       //总变更注册照
}DEV_STATIS;
```

3.6 设备工作属性

```
typedef struct
{
    int    m_TimeGID[DEV_TIMEGROUP_NUMS];   //时间组ID
    int    m_BaseSet;                       //参见设备功能标记定义
    int    m_DoorMangetTime;                //门磁延时时间，单位秒
    int    m_LockTime;                      //电锁持续时间，单位秒
    //验证模式如:DEV_VERIFY_USERID|DEV_VERIFY_CARD|DEV_VERIFY_FACE_11
    int     m_VerifyMode;
    int    m_nWGType;            //韦根协议类型(0输出韦根26,1输出韦根34)
    int m_nWGOutType;            //韦根输出类型(0输出ID，1输出WG内容)
    int m_nWGOutIDType;          //输出ID类型(0输出卡号,1输出用户ID)
    int m_nWGOutContent;         //WG输出内容
    BOOL   m_bWGOutPut;                  //是否WG输出
    DEV_CID      m_szSuperPWD;           //超级密码
    DEV_DEFAULT_RIGHT m_DefaultRight;    //设备默认权限
}DEV_WORKATT;
```

3.7 用户基本信息(处于过时状态)

```
typedef struct
{
    DEV_USER_TYPE m_UserType; //终端设备上的用户类型(普通，管理，超管 -- 0 ,
    1 , 2)
    DEV_CID              m_UserID;          //用户ID
    DEV_DATETIME         m_TypeTime;        //用户类型更改时间
    DEV_DATETIME         m_RegistTime;      //人脸注册时间，即特征采集时
    间，采集或者重新采集要更新此时间
    DEV_DATETIME         m_LastUpdTime;     //最后更新的时间，识别时特征发
    生更新或者重新采集要更新此时间
}DEV_VUSER;
```

3.8 用户详细信息

```
typedef struct
{
    int          m_FeatLen;          //特征大小
    int          m_PicLen;           //照片大小
    int          m_RightID;          //用户权限ID
    DEV_CID    m_ImageSID;           //上传注册照，对应的变更ID，全局唯一
    DEV_CID    m_UserID;             //用户唯一ID,关键字
    DEV_CID    m_Admin;              //人脸注册管理员
```

```
    DEV_CID    m_AppendID;              //卡号或者其他用于1:1的附加身份确认信息
    DEV_CID    m_UserName;              //用户名
    DEV_CID    m_Department;            //部门名称
    DEV_DATETIME m_DeadLineTime;//失效时间
    DEV_DATETIME m_RegistTime;     //人脸注册时间
    DEV_DATETIME m_LastUpdTime; //最后更新的时间
    DEV_DATETIME m_TypeUpdTime;//用户类型变更时间
    char m_UserFlag;                    //用户状态标记
    DEV_USER_TYPE m_UserType;     //终端设备上的用户类型
    char*      m_FeatData;              //特征数据
    char*      m_PicData;               //照片数据
    int        m_bRTUser;               //是否为实时用户(0  非实时用户  1  实时采集
    用户  2实时特征更新用户)
}DEV_USER;
```

3.9 用户基本信息集合(处于过时状态)

```
typedef struct
{
    int m_Count;                  //数组大小
    DEV_VUSER* m_pArray;    //数组起始四肢
}DEV_DL_USER_RANGE;
```

3.10 实时用户加载

```
typedef struct
{
    int          m_LoadFlag;          //DL_DEV_PIC|DL_DEV_USER_FEAT=需要加载
    注册照片和人脸特征
    int          m_ReqSource;         //请求来源,详见485地址解析
    BOOL     m_bApeendID;          //TRUE=卡号, FALSE=工号
    DEV_CID m_ID;                  //加载ID
}DEV_USER_REAL_LOAD;
```

3.11 记录信息

```
typedef struct
{
    DEV_REC_TYPE    m_RecType;            //记录类型
    DEV_REC_SOURCE_TYPE m_Source;    //记录来源(已经过时,不在使用)
    BOOL m_bRealTime;                     //是否为实时记录
    int    m_Score;                       //识别得分
    int    m_PicLen;                      //照片大小
    int    m_VerifyMode;                  //验证模式,例
    如:DEV_VERIFY_USERID|DEV_VERIFY_CARD|DEV_VERIFY_FACE_11
    DEV_CID    m_ChangID;                 //变更记录号
    DEV_CID    m_SerialID;              //识别记录流水号ID
    DEV_CID    m_UserID;                //用户ID,如果为空，表示非法记录
    DEV_CID    m_AppendID;              //卡号或者其他用于1:1的附加身份确认信息
```

```
    DEV_CID    m_UserName;              //用户名
    DEV_CID    m_Department;           //部门名称
    DEV_DATETIME m_RecTime;                 //记录时间
    char    m_ConcretSource;            //记录来源485地址，详见485地址解析
    char* m_PicData;                    //原始JPG图像数据(未经base64编码)
}DEV_RECORD;
```

3.12 日志信息

```
typedef struct
{
    BOOL    m_bRTLog;                  //是否为实时操作日志
    DEV_LOG_TYPE m_LogType;        //日志类型
    DEV_CID m_SerialID;                //流水号ID
    DEV_CID m_Admin;                  //操作员
    DEV_CID m_BeOptUser;              //被操作员
    DEV_DATETIME m_RecTime;        //记录时间
}DEV_LOG;
```

3.13 变更注册照

```
typedef struct    //变更注册照
{
     int                m_PicBytes;     //注册照大小
    BOOL       m_bRTChangeImage;   //是否为实时变更注册照
    DEV_CID    m_UserID;              //用户ID
    DEV_CID    m_SerialID;            //流水号ID
    DEV_CID    m_AppendID;          //卡号或者其他用于1:1的附加身份确认信息
    DEV_CID    m_UserName;          //用户名
    DEV_CID    m_Department;        //部门名称
    DEV_CID    m_Admin;             //人脸注册管理员,标识此用户的人脸特征是哪
    个管理员采集
    DEV_DATETIME m_RecTime;     //记录时间
    char*       m_PicData;              //JPG图像数据(未经base64编码)
}DEV_CHANGEIMAGE;
```

3.14 记录下载区间

```
typedef struct
{
     int    m_Count;          //下载多少条
     BOOL m_bOpenRange; //是否为开区间(true=是，  false=否)
     DEV_CID m_SID;        //从那一条开始，m_SID表示记录精确的流水号  SID组成：
SN + TYPE + DATE + ID = ( 6BYTE + 1BYTE + 8BYTE + 4BYTE + \0)
}DEV_DL_RECORD_RANGE;
```

3.15 记录下载区间集合

```
typedef struct
{
    int m_Count;
    DEV_DL_RECORD_RANGE* m_pRange;
}DEV_DL_RECORD_ARRAY;
```

3.16 时间组

```
typedef struct
{
    int          m_TGID;                    //时间组ID
    DEV_DATETIME m_Start;                   //时间组开始时间
    DEV_DATETIME m_End;                     //时间组结束时间
    DEV_TIMEGROUP_TYPE m_TGType;       //时间组类型
    char         m_CheckFlag;               //时间检测标记例：m_CheckFlag =
    CHECK_TIME|CHECK_WEEK
    char         m_WeekFlag;                //检测星期时，标记那些星期有效。例：
    m_WeekFlag = WEEK_1|WEEK_5
}DEV_TIMEGROUP; //时间组
```

3.17 时间组数组

```
typedef struct
{
    int                  m_nCount;
    DEV_TIMEGROUP* m_pTGArray;
}DEV_TIMEGROUP_ARRAY;/
```

3.18 权限

```
typedef struct
{
     int m_RightID;                         //权限ID
     //时间组ID (m_TimeGID[0]==ANY_TIME)未指定时间组，开门方式不受时间限
     制，任意时间段验证成功执行开门方式.
     int m_TimeGID[DEV_TIMEGROUP_NUMS];
     BOOL m_bHolidyValid;                   //节假日是否有效
     BOOL m_bActionLock;                    //电锁输出
     BOOL m_bActionOutPut;                  //电锁辅助输出
}DEV_RIGHT;
```

3.19 权限数组

```
typedef struct
{
        int                 m_nCount;
        DEV_RIGHT*          m_pRtArray;
}DEV_RIGHT_ARRAY; /
```

3.20 用户组

```
typedef struct
{
    int    m_GroupID;                                    //组ID
    int    m_NormalValid;                                //普通用户中有效用户数
    int    m_ForceValid;                                 //强制用户中有效用户数
    DEV_CID m_NormalUsers[DEV_USER_COMBINS]; //普通用户组合，优先级低
    DEV_CID m_ForceUsers[DEV_USER_COMBINS];    //强制用户组合，优先级高
    BOOL m_bGroupOrder; //组合是否有序  1有序，0无序
}DEV_USERGROUP;
```

3.21用户组数组

```
typedef struct
{
     int          m_nCount;        //数组大小
     DEV_USERGROUP* m_pUGArray; //数组起始地址
}DEV_USERGROUP_ARRAY;
```

3.23 程序更新结构体

```
typedef struct
{
        int         m_Bytes;              //文件大小
        DEV_CID m_FileName;       //文件名
        char        m_TotalFiles;        //需要更新的文件总数
        char        m_FileIndex;        //当前更新到第几个
        char*      m_Buf;               //文件数据缓存首地址
}DEV_APPUPDATE;
```

3.24用户批量操作

```
typedef struct
{
        int            m_nCount;             //数组大小
        DEV_USER* m_pUserArray;    //数组起始地址
}DEV_BATCH_USER;

//批量记录
typedef struct
{
        int                m_nCount;
        DEV_RECORD* m_pRecordArray;
}DEV_BATCH_RECORD;

//批量日志
typedef struct
{
        int            m_nCount;
        DEV_LOG*    m_pLogArray;
}DEV_BATCH_LOG;
```

```
//批量注册照
typedef struct
{
        int                     m_nCount;
        DEV_CHANGEIMAGE*    m_pCImageArray;
}DEV_BATCH_CIMAGE;
```

3.25 设备IO状态

```
//IO设备状态
#define DEV_IO_MODE_NORMAL     0x00     //正常状态
#define DEV_IO_MODE_OPEN       0x01     //常开状态
#define DEV_IO_MODE_CLOSE      0x02     //常关状态
//门磁状态
#define DEV_IO_STATE_CLOSE     0x00      //门磁关
#define DEV_IO_STATE_OPEN      0x01      //门磁开
//执行动作
#define DEV_ACT_IO_OPEN        0x02      //执行打开动作
#define DEV_ACT_IO_OPENEX      0x04      //执行辅助动作
#define DEV_ACT_IO_CLOSE       0x10      //执行关闭动作
#define DEV_ACT_MODE_SET       0x20      //设置IO设备工作模式
#define DEV_ACT_MODE_GET       0x40      //获得IO设备工作模式
#define DEV_ACT_STATE_GET      0x80      //获得IO设备当前状态

typedef struct
{
    char m_Source;        //IO设备485地址
    char m_Action;        //执行动作
    char m_IOMode;        //IO设备当前模式
    char m_IOState;       //IO设备当前状态状态
}DEV_IOCTRL;
```

3.26 远程请求管理端开门

```
typedef struct
{
    int    m_CtxID;              //m_Users==1权限ID, m_Users>1组合ID
    int    m_Users;              //验证用户数目：  0密码开门，1权限开门，>1组合开门
    DEV_IOCTRL m_XOpen; //控制信息
    DEV_CID      m_IDS[DEV_USER_COMBINS*2]; //验证用户数组
}DEV_NOPEN;
```

3.27 IP地址结构

```
typedef struct
{
        unsigned short Port;                //端口
        char                IP_Address[16]; //点分十进制IP地址
}PEERADR;
```

API调用顺序

库初始化与反初始话	
CPM_InitSys	库初始化与反初始话 (必须)
回调函数注册	
CPM_RegDevConnectStatusCB	设备端连接结果通知 (必须)
CPM_RegOperResultNotifyCB	设备操作结果通知 (必须)
模式选择	
CPM_SetMode	选择本地模式或者中转服务模式 (可选)
CPM_CnSrv	中转模式下, 连接中转服务器
设备搜索与连接操作	
CPM_StartDevSerch	设备搜索获得指定组播下所有在线设备的IP
CPM_CNDev	与指定IP人脸机建立网络连接
CPM_DCNDev	断开连接
CPM_DCNAllDev	断开所有连接
设备相关信息设置与获取	
CPM_ULDevAuth	远程客户端身份验证
CPM_ULDevTime	设置设备系统时间
CPM_DLDevTime	获取设备系统时间
CPM_DLDevBaseInfo	获取设备基本信息
CPM_DLDevStatisInfo	获取设备统计信息
CPM_DLDevWorkAttInfo	获取设备工作属性
CPM_ULDevWorkAttInfo	设置设备工作属性
设备用户操作	
CPM_ULUser	上传用户
CPM_ULRealTimeUser	上传实时用户
CPM_ULRealTimeUserCap	远程采集用户
CPM_DELUser	删除某用户
CPM_DLSingleUser	下载某一用户
CPM_DLAllUser	下载所有用户
CPM_DLSegTimeUser	下载某一时间段内的注册用户
设备用户批量操作	
CPM_ULUserEx	批量上传用户 (推荐)
CPM_DLUserEx	批量下载用户 (推荐)
CPM_DELUserEx	批量删除用户 (推荐)
设备验证记录操作	
CPM_DLAllIdentifyRecord	下载所有识别记录
CPM_DLSegTimeIdentifyRecord	下载指定时间段识别记录
CPM_DLRangeRec	下载区间识别记录
CPM_DLAllAlarmRecord	下载所有报警记录
CPM_DLSegTimeAlarmRecord	下载指定时间段报警记录
CPM_DLRangeAlarm	下载区间报警记录
设备日志操作	
CPM_DLLog	下载所有日志
CPM_DLRangeLog	下载区间日志
设备变更注册照操作	
CPM_DLChangeImage	下载所有变更注册照
CPM_DLRangeCImage	下载区间变更注册照

设备时间组操作	
CPM_ULTimeGroup	上传时间组
CPM_DLTimeGroup	下载时间组
CPM_DELTimeGroup	删除时间组
设备权限操作	
CPM_ULRight	上传权限
CPM_DLRight	下载权限
CPM_DELRight	删除权限
设备用户组操作	
CPM_ULUserGroup	上传用户组
CPM_DLUserGroup	下载用户组
CPM_DELUserGroup	删除用户组
设备门禁操作	
CPM_ULOpenDoorState	设置人脸门(常开，常闭，常规)状态
CPM_DLOpenDoorState	获得人脸门，门状态
CPM_ULRemoteOpenDoor	远程开/关门
CPM_IOCtrl	远程控制/设置，IO设备 (推荐)
设备程序更新与数据格式化操作	
CPM_ULFormat	格式化设备
CPM_ULUpdate	设备固件更新
CPM_RebootDev	远程重启人脸机
记录批量下载(验证，报警，变更)	
CPM_DLRecEx	批量下载记录 (推荐)
远程视频	
CPM_StartVideo	开启远程视频
CPM_StopVideo	停止远程视频

API函数详解

1. 回调函数定义（重要）

/**/
/*功　　能 设备或者中转服务器连接状态回调函数
/*参　　数 cszDevAddr 人脸机IP或者中转服务器IP
　　eCNStatus　连接状态
　　pvContext　应用上下文
/*说　　明 当连接人脸机和中转服务器发生，连接成功，连接失败，异常断开或者搜索到设备时，通过该回调函数给出，eCNStatus为连接状态DEV_CONNECT_STATUS
/**/
typedef void (*DevConnectStatus)(const char* cszDevAddr,
DEV_CONNECT_STATUS eCNStatus, void* pvContext);

/**/
功　　能 查询设备所属中转服务器，返回设备所属中转服务器IP地址
参　　数
　　cszDevAddr　人脸机IP
　　srvAddr　　需要返回的中转服务器IP
　　pvContext　　上下文情景参数
说　　明　当存在多个中转服务器时，连接设备会通过该回调函数向外部查询该设备

所属中转服务器IP。

/***/

typedef BOOL (CPMDEV_CALL *QuerySrvCB)(const char* cszDevAddr, PEERADR& srvAddr, void* pvContext);

/***/

/*功　　能　设备操作结果回调函数定义

/*参　　数　cszDevAddr　设备地址

eType　操作类型

eFlag　操作标识

pvContent　操作结果内容

nSeq　操作流水号

eResult　操作结果

pvContext　应用层上下文

/*说　　明　上层应用收到此消息后即可知道前一次操作是否成功，eType、eFlag、eResult 共同决定 pvContent 内容　当 eResult=OPER_SUCCESS && eFlag=DEV_OPER_DOWNLOAD时 pvContent根据eType决定具体数据内容，否则 pvContent=NULL。

DEV_AUTH_OPER　pvContent=NULL　客户端身份验证反馈

DEV_AUTH_SET_OPER　pvContent=NULL　客户端连接用户名密码设置成功反馈

DEV_REGION_OPER　pvContent=NULL

DEV_FORMAT_OPER　pvContent=NULL

DEV_UPDATE_OPER　pvContent=NULL

DEV_SYS_TIME_OPER　pvContent=DEV_DATETIME* or = NULL

DEV_BASEINFO_OPER　pvContent=DEV_BASEINFO* or = NULL

DEV_STATISINFO_OPER　pvContent=DEV_STATIS* or = NULL

DEV_WORKATT_OPER　pvContent=DEV_WORKATT* or = NULL

DEV_USER_OPER　pvContent=DEV_USER* or = NULL

DEV_USER_RT_OPER　pvContent=DEV_USER_REAL_LOAD*

DEV_USER_RANGE_OPER　pvContent=DEV_DL_USER_RANGE* or = NULL

DEV_RECORD_OPER　pvContent=DEV_RECORD* or = NULL

DEV_RECORD_RANGE_OPER　pvContent=DEV_DL_RECORD_RANGE* or = NULL

DEV_LOG_OPER　pvContent=DEV_LOG* or = NULL

DEV_LOG_RANGE_OPER　pvContent=DEV_DL_RECORD_RANGE* or = NULL

DEV_CHANGEIMAGE_OPER　pvContent=DEV_CHANGEIMAGE* or = NULL

DEV_CIMAGE_RANGE_OPER　pvContent=DEV_DL_RECORD_RANGE* or = NULL

DEV_TIME_GROUP_OPER　pvContent=DEV_TIMEGROUP_ARRAY* or = NULL

DEV_RIGHT_OPER　pvContent=DEV_RIGHT_ARRAY* or = NULL

DEV_USERGROUP_OPER　pvContent=DEV_USERGROUP_ARRAY* or = NULL

DEV_DOOR_STATE_OPER　pvContent=DEV_DOOR_STATE* or = NULL

DEV_NOPEN_OPER　pvContent=DEV_NOPEN*设备请求管理端远程开门

DEV_IOCTRL_OPER　pvContent=DEV_IOCTRL*　IO控制信息反馈

CLI_AUTH_OPER　pvContent=AUTH*　nSeq代表client端口

如果调用API时指定的序列号nSeq的值与回调函数参数nSeq 的值相等，则表示该回复和之前操作是对应的。请确保nSeq的唯一性，0序号为系统保留，不能使用。

/***/

typedef void (*DevOperResultNotify)(

const char* cszDevAddr,

```
DEV_OPER_TYPE eType, DEV_OPER_FLAG eFlag,
void* pvContent, int nSeq,
DEV_OPER_RESULT eResult, void* pvContext );
```

2. 系统初始化/反初始化

```
/*****************************************************************************/
/*功      能  初始化/反初始化系统
/*参      数  bFlag(true=初始化   false=反初始化)
/*说      明  使用之前调用此接口初始化SDK，退出之前反初始化SDK释放资源.
/*****************************************************************************/
BOOL CPM_InitSys(BOOL bFlag)

/*****************************************************************************/
/*功      能   注册设备和中转服务器连接状态回调函数
/*参      数   pfnCNNotify  回调函数指针，参见回调函数定义
               pvContext      应用上下文
/*说      明   调用此接口后，当对设备或中转服务器进行网络连接时，连接结果会通过回调函数通知调用者。在连接人脸机或中转服务器之前必须设置该回调函数，在反初始化之前设置为NULL
/*****************************************************************************/
void CPM_RegDevConnectStatusCB(DevConnectStatus pfnCNNotify, void* pvContext);

/*****************************************************************************/
/*功      能  注册设备操作结果通知
/*参      数  pfnOperNotify     回调函数指针，参见回调函数定义
               pvContext         应用上下文
 /*说      明  调用此接口后，对设备的所有操作或设备主动向客户端发送所有消息，
 都将通过注册的回调函数通知外部，具体数据解析请参考回调函数说明。
/*****************************************************************************/
void CPM_RegOperResultNotifyCB(DevOperResultNotify pfnNofity,   void*pvContext)
```

重要说明：

1.使用SDK之前，必须调用CPM_InitSys函数初始化SDK

2.所有设备连接或断开的通知通过CPM_RegDevConnectStatusCB注册的回调函数通知外部

3.所有操作结果反馈(例如下载用户，记录)，统一通过CPM_RegOperResultNotifyCB注册的回调函数通知外部。所以必需在调用其他操作函数之前注册以上两个回调函数

4. CPM_CNDev尝试连接指定人脸机，结果通过CPM_RegDevConnectStatusCB注册的回调函数给出。可通过CPM_StartDevSerch函数获得所有在线设备的IP，然后通过IP连接设备

5.反初始化之前，先将以上两个回调函数设置为NULL

6.注意事项：DevOperResultNotify回调中不能处理太复杂，处理时间不能超过1秒，否则会阻塞其他操作

```
/*************************************************************************/ /*
功      能 设置整个模块的工作模式（本地或者中转）
/*参      数
          srvMode      工作模式 TRUE 中转模式， FALSE本地模式
          pfn          中转服务查询回调函数，参见回调函数定义
          pvContext    应用上下文
/*说      明 不调用该函数，默认为本地模式。设置为中转模式时，当srvMode设置为NULL时，默认采用第一个中转服务器。因此，当只存在一个中转服务器时，可设置为NULL。
/*************************************************************************/
CPMDEV_API BOOL CPM_SetMode( BOOL srvMode, QuerySrvCB pfn,   void* pvContext );

/*************************************************************************/
功      能 连接到中转服务器，仅在中转模式下调用。
参      数
          bFlag      true=连接中转服务器, false=断开与中转服务器的连接
          ip         中转服务器IP地址
          name       client验证用户名
          psw        client验证密码
说      明 中转服务器连接成功返回TRUE，否则FALSE。身份验证成功与否通过CPM_RegDevConnectStatusCB注册的回调函数通知外部. 成功 eCNStatus == DEV_CONNECT_SUCCESS ， 失败eCNStatus   == DEV_CONNECT_FAILUE
/*************************************************************************/
CPMDEV_API BOOL CPMDEV_CALL CPM_CnSrv( BOOL bFlag, const char* ip, const char* name, const char* psw );
```

3．和人脸机建立连接

```
/*************************************************************************/
功      能   与人脸机建立连接
参      数   cszDevAddr 人脸机地址
说      明   成功 eCNStatus == DEV_CONNECT_SUCCESS ， 失败eCNStatus   == DEV_CONNECT_FAILUE，通过注册回调函数通知外部。
/*************************************************************************/
BOOL CPM_CNDev( const char* cszDevAddr );

/*************************************************************************/
功      能   管理端身份合法性验证，或者重新设置验证用户名，密码。
参      数
          cszDevAddr   人脸机地址
          cName        用户名 （默认: admin）
          cPws         密码      (默认:201031)
          nFlag        1=验证 2=修改
          nSeq         操作序列号，不能为0
说      明   对人脸机操作之前需调用此接口进行验证是否为合法管理端，如果未修改原始用户名和密码，通过用户名：admin   密码：201031进行验证。验证成功后方可修改原始用户名和密码。客户端身份验证必须在连接建立后的30秒内完成，否则连接
```

自动断开。 DevOperResultNotify回调参数对应类型：eType = DEV_AUTH_OPER
eFlag = DEV_OPER_UPLOAD，pvContent = NULL

```
/*************************************************************************/
BOOL CPM_ULDevAuth( const char* cszDevAddr, const DEV_CID& cName,
const DEV_CID& cPws, int nFlag,   int nSeq = -1);
```

4. 断开设备连接

```
/*************************************************************************/
```

功　　能 断开人脸机连接
参　　数 cszDevAddr 人脸机地址
说　　明 结果以DevConnectStatus回调通知, eCNStatus == DEV_CONNECT_CUT

```
/*************************************************************************/
void CPM_DCNDev( const char* cszDevAddr );
```

5. 自动搜索设备

```
/*************************************************************************/
```

功　　能　启动或停止搜索指定组播下的所有在线设备
参　　数
　　bFlag　　(true=启动　false=停止)
　　cszRgnAddr　设备所在区域地址
说　　明　调用此接口用以获得指定组播地址下所有在线设备的IP地址。如果是搜索设备，当搜索到有在线设备时通过DevConnectStatus回调通知，此时 eCNStatus == DEV_CONNECT_NONE ，cszDevAddr==设备IP，如有N台设备在线，该回调函数调用N次。

```
/*************************************************************************/
BOOL CPM_StartDevSerch(BOOL bFlag, const char* cszRgnAddr = DEV_RGN_ADDR)
```

6. 上传/下载/删除 时间组

```
/*************************************************************************/
```

功　　能　上传时间组
参　　数
　　cszDevAddr 设备地址
　　cTGArray　时间组数组
　　nSeq　　操作序列
说　　明 DevOperResultNotify回调参数对应类型：
eType = DEV_TIME_GROUP_OPER
eFlag = DEV_OPER_UPLOAD ， pvContent = NULL

```
/*************************************************************************/
BOOL CPM_ULTimeGroup(const char* cszDevAddr,
const DEV_TIMEGROUP_ARRAY& cTGArray, int nSeq = -1);

/*************************************************************************/
```

功　　能　下载时间组
参　　数
　　dev_addr　设备地址
　　cTGArray　时间组数组
　　eType　　时间组类型

nSeq 操作序列
说 明
cTGArray.m_nCount=0时表示下载eType类型的所有时间组
DevOperResultNotify回调参数对应类型：eType = DEV_TIME_GROUP_OPER
eFlag = DEV_OPER_DOWNLOAD ， pvContent = DEV_TIMEGROUP_ARRAY*
/***/
BOOL CPM_DLTimeGroup(const char* cszDevAddr,
const DEV_TIMEGROUP_ARRAY& cTGArray,
DEV_TIMEGROUP_TYPE eType, int nSeq = -1);

/***/
功 能 删除时间组
参 数
cszDevAddr 设备地址
cTGArray 时间组
eType 时间组类型
nSeq 操作序列
说 明
eType.m_nCount=0时表示删除tg_type类型所有时间组
DevOperResultNotify回调参数对应类型：eType = DEV_TIME_GROUP_OPER
eFlag = DEV_OPER_DELETE ， pvContent = NULL
/***/
BOOL CPM_DELTimeGroup(
const char*cszDevAddr,
const DEV_TIMEGROUP_ARRAY& cTGArray,
DEV_TIMEGROUP_TYPE eType, int nSeq = -1);
重要说明：时间组ID必须大于0

7. 上传/下载/删除 权限

/***/
功 能 上传权限
参 数
cszDevAddr 设备地址
cRTArray 权限数组
nSeq 操作序列
说 明 DevOperResultNotify回调参数对应类型：eType = DEV_RIGHT_OPER
eFlag = DEV_OPER_UPLOAD，pvContent = NULL
/***/
BOOL CPM_ULRight(
const char* cszDevAddr,
const DEV_RIGHT_ARRAY& cRTArray,
int nSeq = -1);

/***/
功 能 下载权限
参 数
cszDevAddr 设备地址

cRTArray　权限数组
nSeq　操作序列
说　　明
当rt_array.m_nCount=0时表示下载所有权限
DevOperResultNotify回调参数对应类型：eType = DEV_RIGHT_OPER
eFlag = DEV_OPER_DOWNLOAD，pvContent = DEV_RIGHT_ARRAY*

```
/*************************************************************************/
BOOL CPM_DLRight(
const char* cszDevAddr,
const DEV_RIGHT_ARRAY& cRTArray,
int nSeq = -1);

/*************************************************************************/
```

功　　能　删除权限
参　　数
cszDevAddr　设备地址
cRTArray　　权限数组
nSeq　　操作序列
说　　明　当rt_array.m_nCount=0时表示删除所有权限
DevOperResultNotify回调参数对应类型：eType = DEV_RIGHT_OPER
eFlag = DEV_OPER_DELETE，pvContent = NULL

```
/*************************************************************************/
BOOL CPM_DELRight(
const char* cszDevAddr,
const DEV_RIGHT_ARRAY& cRTArray,
int nSeq = -1);
```

重要说明：权限ID必须大于0

8．上传/下载/删除 组合开门用户组

```
/*************************************************************************/
```

功　　能　上传用户组
参　　数
cszDevAddr 设备地址
cUGrray　　用户组数组
nSeq　　操作序列
说　　明　DevOperResultNotify回调参数对应类型：
eType = DEV_USERGROUP_OPER
eFlag = DEV_OPER_UPLOAD，pvContent = NULL

```
/*************************************************************************/
BOOL CPM_ULUserGroup(
const char* cszDevAddr,
const DEV_USERGROUP_ARRAY& cUGrray,
int nSeq = -1);

/*************************************************************************/
```

功　　能　下载用户组
参　　数
cszDevAddr 设备地址

ug_array 用户组数组
nSeq 操作序列
说 明
当rt_array.m_nCount=0时表示下载所有用户组
DevOperResultNotify回调参数对应类型：eType = DEV_USERGROUP_OPER
eFlag = DEV_OPER_DOWNLOAD，pvContent = DEV_USERGROUP_ARRAY*

```
/*************************************************************************/
BOOL CPM_DLUserGroup(
const char* cszDevAddr,
const DEV_USERGROUP_ARRAY& cUGrray,
int nSeq = -1);

/*************************************************************************/
```

功 能 删除用户组
参 数
cszDevAddr 设备地址
ug_array 用户组数组
nSeq 操作序列
说 明
当rt_array.m_nCount=0时表示删除所有用户组
DevOperResultNotify回调参数对应类型：eType = DEV_USERGROUP_OPER
eFlag = DEV_OPER_DELETE，pvContent = NULL

```
/*************************************************************************/
BOOL CPM_DELUserGroup(
    const char* cszDevAddr,
    const DEV_USERGROUP_ARRAY& cUGrray,
    int nSeq = -1);
```

重要说明：用户组ID必须大于0

9. 上传/下载/删除 用户

```
/*************************************************************************/
```

功 能 下载所有用户文字信息
参 数
cszDevAddr 设备地址
nSeq 操作序列号
说 明 不包括用户的注册照与人脸库信息
DevOperResultNotify回调参数对应类型：eType = DEV_USER_RANGE_OPER
eFlag = DEV_OPER_DOWNLOAD，pvContent = DEV_BATCH_USER*

```
/*************************************************************************/
BOOL CPM_DLAllUser(const char* cszDevAddr, int nSeq = -1);
```

```
/*************************************************************************/
功      能   上传指定用户
参      数
        cszDevAddr  设备地址
        cUser       用户信息结构体
        nSeq        操作序列号
说      明
用户无人脸库时，DEV_USER中的 m_FeatLen = 0，  m_FeatData = NULL
用户无注册照时，DEV_USER中的 m_PicLen= 0，  m_PicData = NULL
DevOperResultNotify回调参数对应类型：eType = DEV_USER_OPER
eFlag = DEV_OPER_UPLOAD，pvContent= NULL
/*************************************************************************/
BOOL CPM_ULUser(const char* cszDevAddr, const DEV_USER& cUser, int nSeq = -1) ;

/*************************************************************************/
功      能   删除指定用户
参      数
        cszDevAddr  设备地址
        cID         用户编号
        nSeq        操作序列号
说      明   删除成功与否，通过操作回调函数给出结果
DevOperResultNotify回调参数对应类型：
eType = DEV_USER_OPER
eFlag = DEV_OPER_DELETE，
pvContent =   NULL
/*************************************************************************/
BOOL CPM_DELUser(const char* cszDevAddr, const DEV_CID& cID, int nSeq = -1);

/*************************************************************************/
功      能   下载指定用户
参      数
        cszDevAddr  设备地址
        cID         用户ID
        nFlag       DL_DEV_USER_PIC  需下载用户照片
                    DL_DEV_USER_FEAT  需要下载人脸特征
                    DL_DEV_USER_PIC|DL_DEV_USER_FEAT=两者
                    为0默认下载文字信息

        nSeq        操作序列号
说      明   DevOperResultNotify回调参数对应类型：
eType = DEV_USER_OPER
eFlag = DEV_OPER_DOWNLOAD，
pvContent =   DEV_USER*
/*************************************************************************/
BOOL CPM_DLSingleUser(const char* cszDevAddr, const DEV_CID& cID, int nFlag,
int nSeq = -1);
```

```
/*************************************************************************/
功      能 批量上传用户
参      数
        cszDevAddr 设备地址
        cUsers      批量用户数据
        nFlag       DL_DEV_PIC                  //照片
                    DL_DEV_USER_FEAT      //人脸特征
                    DL_DEV_USER_TEXT      //用户文字信息
说      明: 批量上传用户必须包含用户文字信息，组合如下
nFlag = DL_DEV_USER_TEXT 上传文字信息
nFlag = DL_DEV_USER_TEXT|DL_DEV_PIC   文字+注册照
nFlag = DL_DEV_USER_TEXT|DL_DEV_USER_FEAT 文字+人脸库
nFlag = DL_DEV_USER_TEXT|DL_DEV_PIC|DL_DEV_USER_FEAT文字+注册照+人脸库
DevOperResultNotify回调参数对应类型：
eType = DEV_USER_EX_OPER
eFlag = DEV_OPER_UPLOAD ，
pvContent = DEV_BATCH_USER*。
在OPER_SUCCESS 情况下，批量上传的用户个数与返回的用户个数一致。否则设备保存失败。批量上传用户数设定在5个以下(和网络带宽，主机处理速度有关)。
/*************************************************************************/
CPMDEV_API BOOL CPM_ULUserEx(
const char* cszDevAddr,
const DEV_BATCH_USER& cUsers,
int nFlag, int nSeq = -1 );

/*************************************************************************/
功      能 批量下载用户
参      数
        cszDevAddr      设备地址
        pUserIDArray    用户ID数组首地址
        nIDCount        数组元素个数
        nFlag           同CPM_ULUserEx
说      明 DevOperResultNotify回调参数对应类型：
eType = DEV_USER_EX_OPER
eFlag = DEV_OPER_DOWNLOAD ，
pvContent= DEV_BATCH_USER*。
OPER_SUCCESS 情况下，如果批量下载的用户数与要求下载的用户数不一致，可以通过 CPM_DLAllUser(const char* cszDevAddr, int nSeq = -1) 来确认用户是否真的存在于设备上。 批量下载用户数设定在5个以下(和网络带宽，主机处理速度有关)。
/*************************************************************************/
CPMDEV_API BOOL CPM_DLUserEx(
const char* cszDevAddr,
DEV_CID* pUserIDArray,
int nIDCount, int nFlag, int nSeq = -1 );
```

/***/
功　　能 批量删除用户
参　　数
　　cszDevAddr　　设备地址
　　pUserIDArray　待删除用户的编号数组
　　nIDCount　　数组元素个数
说　　明 DevOperResultNotify回调参数对应类型：
eType = DEV_USER_EX_OPER
eFlag = DEV_OPER_DELETE，
pvContent= DEV_BATCH_USER*。
在OPER_SUCCESS 情况下，批量删除的用户个数与返回的用户个数不一致，可以通过CPM_DLAllUser(const char* cszDevAddr, int nSeq = -1) 来确认用户是否真的存在于设备上。单次批量删除用户数设定在10个以下为好。
/***/
CPMDEV_API BOOL CPM_DELUserEx(const char* cszDevAddr, DEV_CID* pUserIDArray, int nIDCount, int nSeq = -1);
重要说明：
1．DEV_USER结构中的m_ImageSID描述如下：XYYYYY5ZZZZ…，第7位必须是数字5为变更注册照类型，如果第一位为数字，在采集或重新采集时，用户的注册照将设备所取的现场照覆盖。如果第一位为字符，则不会发生改变，为第一次传入的照片。
2．批量接口：批量用户处理接口为单个用户处理的扩展接口。在处理大量用户上传，删除，下载操作时性能有很大的提高。
3．用户权限如果大于0，识别成功后就执行用户权限，如果用户权限小于0说明没有设置权限，识别成功后执行设备权限。设备相关权限参考DEV_DEFAULT_RIGHT。

10．变更注册照，识别记录，报警记录
/***/
功　　能　获取变更注册照记录区间
参　　数
　　cszDevAddr 设备地址
　　nSeq　　操作序列
说　　明
DevOperResultNotify回调参数对应类型：
eType = DEV_CHANGEIMAGE_OPER
eFlag = DEV_OPER_DOWNLOAD，
pvContent = DEV_DL_RECORD_ARRAY*
参考DEV_DL_RECORD_ARRAY结构，变更注册照只有一个区间
/***/
BOOL CPM_DLChangeImage(const char* cszDevAddr, int nSeq = -1);

/***/
功　　能　下载指定区间的变更注册照记录或者查询区间数据的具体大小
参　　数
　　cszDevAddr　设备地址
　　cRange　　区间，参考区间定义
　　flag　　DL_DEV_RECORD_PIC=需要下载记录照片
　　bKnowRange　true=具体数据，　false=区间集合

nSeq 操作序列
说 明 当bKnowRange 为true该函数下载具体的数据，为false时查询指定区间在设备上的具体大小。 DevOperResultNotify回调参数对应类型：
下载区间时
pvContent = DEV_DL_RECORD_ARRAY*
cRange.m_SID = 指定SN
cRange.m_Count = -1;
cRange.m_bOpenRange = TRUE;
bKnowRange = FALSE，flag = 1（无效参数）
下载具体记录时
DevOperResultNotify回调参数对应类型：
pvContent=DEV_RECORD*
/***/
BOOL CPM_DLRangeCImage(const char* cszDevAddr,
const DEV_DL_RECORD_RANGE& cRange , int nFlag,
BOOL bKnowRange = FALSE, int nSeq = -1);

/***/
功 能 下载识别记录区间
参 数
cszDevAddr 设备地址
nSeq 操作序列
说 明 DevOperResultNotify回调参数对应类型：
eType = DEV_RECORD_OPER
eFlag = DEV_OPER_DOWNLOAD
pvContent=DEV_DL_RECORD_ARRAY*
参考DEV_DL_RECORD_ARRAY结构，一天为一个区间，该函数返回识别记录的所有区间。
/***/
BOOL CPM_DLAllIdentifyRecord(const char* cszDevAddr, int nSeq = -1);

/***/
功 能 下载报警记录区间
参 数
cszDevAddr 设备地址
nSeq 操作序列
说 明 DevOperResultNotify回调参数对应类型：
eType = DEV_ALARM_OPER
eFlag = DEV_OPER_DOWNLOAD，
pvContent = DEV_DL_RECORD_ARRAY*
参考DEV_DL_RECORD_ARRAY结构，报警记录只有一个区间
/***/
BOOL CPM_DLAllAlarmRecord(const char* cszDevAddr, int nSeq = -1);

/***/
功　　能 获取区间识别记录/报警记录指定ID的后续区间
参　　数
　　cszDevAddr　设备地址
　　cRange ID　　区间
　　flag　　DL_DEV_RECORD_PIC=需要下载记录照片
　　bKnowRange　true=具体数据， false=区间集合
　　nSeq　操作序列
说　　明 当bKnowRange 为true该函数下载具体的数据，为false时查询指定区间在设备上的具体大小。 DevOperResultNotify回调参数对应类型：
下载区间时
pvContent = DEV_DL_RECORD_ARRAY*
cRange.m_SID　= 指定SN
cRange.m_Count = -1;
cRange.m_bOpenRange = TRUE;
bKnowRange = FALSE，flag = 1（无效参数）
下载具体记录时
DevOperResultNotify回调参数对应类型：
pvContent=DEV_RECORD*
/***/
CPMDEV_API BOOL CPM_DLRangeRec(
const char* cszDevAddr,
const DEV_DL_RECORD_RANGE& cRange ,
int nFlag, BOOL bKnowRange = FALSE, int nSeq = -1);

/***/
功　　能　批量下载记录 (推荐使用)
参　　数　cszDevAddr 设备地址
　　cRange　　ID区间
说　　明　下载指定记录ID后续cRange区间大小的记录，包括验证记录、报警记录，变更注册照。DevOperResultNotify回调参数对应类型：
变更注册照：pvContent　=　DEV_BATCH_CIMAGE *
验证记录：　pvContent　=　DEV_BATCH_RECORD *
报警记录：　pvContent　=　DEV_BATCH_RECORD *
批量下载记录限定在50条以内。设备通过ID来区分验证记录、报警记录，变更注册照
/***/
CPMDEV_API BOOL CPM_DLRecEx(　const char* cszDevAddr,
const DEV_DL_RECORD_RANGE& cRange, int nSeq = -1);
重要说明：
1．必需保存下载下来的记录流水号m_SerialID，用最后的流水号作为下次记录下载的起始位置，这样可以只下载最近产生的记录。
2．结构体DEV_DL_RECORD_RANGE
m_bOpenRange = TRUE 半开区间下载，说明向设备请求下载从m_SID开始但不包括m_SID的后续m_Count条记录。
m_bOpenRange = FALSE 半闭区间下载，说明向设备请求下载从m_SID开始包括m_SID在内的后续m_Count条记录。

11. 获得/设置人脸机工作属性

```
/*************************************************************************/
功      能   获取人脸机工作属性
参      数
        cszDevAddr  人脸机地址
        nSeq        操作序列号
说      明    工作属性DEV_WORKATT通过回调函数给出。
/*************************************************************************/
BOOL CPM_DLDevWorkAttInfo(const char* cszDevAddr, int nSeq = -1);

/*************************************************************************/
功      能   设置设备工作属性信息
参      数
        cszDevAddr  设备地址
        cWorkAtt    工作属性结构体
        nSeq        操作序列号
/*************************************************************************/
BOOL CPM_ULDevWorkAttInfo(const char* cszDevAddr, const DEV_WORKATT&
cWorkAtt, int nSeq = -1);
```

重要说明：人脸机的工作属性的设置非常重要，可配置人脸机大部分工作状态。

DEV_WORKATT:: m_BaseSet 用于使能各种功能，具体参考.h文件的2.0，3.0的宏定义，所有预定义功能可以组合使用。例如开启实时动态记录和采集用户同步：DEV_WORKATT:: m_BaseSet |= DEV_REALTIME_RECORD| DEV_REALTIME_USERSEND。

DEV_WORKATT:: m_LockTime 设置电锁的持续时间
DEV_WORKATT:: m_DoorMangetTime 人脸门的门磁超时时间，用于门磁报警
DEV_WORKATT::m_TimeGID 配置人脸机工作时间段，暂未使用

开门超级密码，需要使能DEV_WORKATT:: m_BaseSet |= DEV_SUPER_PASSWORD
DEV_WORKATT ::m_szSuperPWD 默认密码888888

韦根信号输出
DEV_WORKATT:: m_bWGOutPut 是否输出韦根信号
DEV_WORKATT:: m_nWGType 韦根信号类型0==wg_26 , 1==wg_34
DEV_WORKATT:: m_nWGOutType 韦根数据输出类型
0 根据m_nWGOutIDType的指示输出
1 根据m_nWGOutContent的内容输出
DEV_WORKATT:: m_nWGOutIDType 0用户标号，1用户卡号
DEV_WORKATT:: m_nWGOutContent 用户自定义内容

DEV_WORKATT ::m_DefaultRight 设备默认权限，权限判定逻辑为，先判定用户是否具有指定权限，如果用户根本没有配置过权限，则将按照设备m_DefaultRigh设定的权限执行。
DEV_DEFAULT_NO_RIGHT 人脸机默认不开门
DEV_DEFAULT_SINGLE_OPEN 人脸机默认开门

DEV_WORKATT::m_VerifyMode 设置人脸机的验证模式，模式可以组合使用例如：
a. 卡+人脸识别 DEV_VERIFY_CARD| DEV_VERIFY_FACE_11
b. ID/ 卡 + 人 脸 DEV_VERIFY_USERID| DEV_VERIFY_CARD| DEV_VERIFY_FACE_11
验证模式详细描述(不支持直接密码验证开门模式)：
刷卡模式：DEV_WORKATT::m_BaseSet==DEV_VERIFY_CARD，人脸机退化为刷卡机
1:1 模式：通过刷卡(DEV_VERIFY_CARD)，输入工号(DEV_VERIFY_USERID)等方式定位出用户身份，然后在此先验知识上进行人脸识别.如果输入的卡号或者工号不再当前设备中，若条件 DEV_WORKATT::m_BaseSet& DEV_REALTIME_USERLOAD 成立，将尝试从网络实时加载对应用户信息。
1:N模式：DEV _WORKATT::m_BaseSet|DEV_VERIFY_FACE_1N为真，则设备中的所有有特征的用户N加入内存人脸库(N <= Limit1N)，进行无先验知识的人脸识别。如果 N> Limit1N， 人脸机工作模式自动转变为 1:1 模式 (DEV_VERIFY_USERID| DEV_VERIFY_CARD| DEV_VERIFY_FACE_11)。
混合模式：DEV_USER:: m_UserFlag|&DEV_USER_MIX1N为真的用户进行1:N验证。

任何用户(包括存在于1:N中的用户)只要工作模式标记了DEV_VERIFY_CARD或DEV_VERIFY_USERID都可以进行1:1识别。

12. 设置/获得人脸机时间

```
/*************************************************************************/
功      能 设置设备系统时间
参      数
        cszDevAddr 设备地址
        cSysTime 时间信息
说      明 DevOperResultNotify回调参数对应类型：
eType = DEV_SYS_TIME_OPER
eFlag = DEV_OPER_UPLOAD ，
pvContent = NULL
/*************************************************************************/
BOOL CPM_ULDevTime(const char* cszDevAddr, const DEV_DATETIME& cSysTime, int nSeq = -1);

/*************************************************************************/
功      能 获取设备系统时间
参      数 cszDevAddr 设备地址
说      明 DevOperResultNotify回调参数对应类型：
eType = DEV_SYS_TIME_OPER
eFlag = DEV_OPER_DOWNLOAD，
pvContent   = DEV_DATETIME*
/*************************************************************************/
BOOL CPM_DLDevTime(const char* cszDevAddr, int nSeq = -1);
```

13. 获得人脸机基本信息

```
/*************************************************************************/
功　　能 获取设备基本信息
参　　数
　　cszDevAddr 设备地址
　　nSeq　　操作序列号
说　　明 操作结果通过回调函数返回DEV_BASEINFO结构体
/*************************************************************************/
BOOL CPM_DLDevBaseInfo(const char* cszDevAddr, int nSeq =-1 );
```

14. 获得人脸机数据统计信息

```
/*************************************************************************/
功　　能　获取设备统计信息
参　　数
　　cszDevAddr 设备地址
　　nSeq　　操作序列号
说　　明　操作结果通过回调函数返回DEV_STATIS结构体
/*************************************************************************/
BOOL CPM_DLDevStatisInfo(const char* cszDevAddr,int nSeq = -1);
```

15. 人脸机格式化

```
/*************************************************************************/
功　　能　格式化设备
参　　数
　　cszDevAddr 设备地址
　　nSeq　　操作序列
说　　明　格式化操作将删除人脸机上所有用户资料和记录数据，当数据量比较大时格式化操作完成时间可达30秒
/*************************************************************************/
BOOL CPM_ULFormat( const char* cszDevAddr, int nSeq = -1);
```

16.人脸门远程相关操作

```
/*************************************************************************/
功　　能　远程开/关门
参　　数
　　cszDevAddr 设备地址
　　bEOpen　　true=开门，false=关门
　　nSeq　　操作序列
说　　明　　管理端通过该操作远程开/关门，此方式开门不受人脸机权限限制.
/*************************************************************************/
BOOL CPM_ULRemoteOpenDoor( const char* cszDevAddr, BOOL bOpen, int nSeq = -1);

/*************************************************************************/
功　　能 获取门状态
参　　数 cszDevAddr 设备地址
说　　明 通过回调函数返回人脸门的状态(正常，常开，常闭)
```

/***/
CPMDEV_API BOOL CPM_DLOpenDoorState(const char* cszDevAddr, int nSeq = -1);

/***/
功　　能 设置开关门状态
参　　数 cszDevAddr 设备地址
DEV_DOOR_STATE　DEV_DOOR_NOMAL或
DEV_DOOR_OPEN　或
DEV_DOOR_CLOSE
说　　明 可设置人脸门为正常工作状态，常开，常闭
/***/
CPMDEV_API BOOL CPM_ULOpenDoorState(const char* cszDevAddr, DEV_DOOR_STATE eState, int nSeq = -1);
重要说明: 常开，常闭，正常，为门的三种工作状态。当处于常闭状态下，人脸识别开门或者远程开门都将失效，必须将门设置为正常工作状态才能响应开门指令.

/***/ /*
功　　能 IO设备控制 (推荐使用)
参　　数
cszDevAddr 设备地址
DEV_IOCTRL 控制信息
nSeq　　命令执行序列号
说　　明 DEV_IOCTRL:m_Source指定要做操作的端口，DEV_IOCTRL:m_Action要执行的动作，该函数是远程开关门，常开常闭，打开辅助输出等操作的通用函数。
打开门1，m_Source = 0x1f　m_Action = DEV_ACT_IO_OPEN
打辅助1，m_Source = 0x15　m_Action = DEV_ACT_IO_OPEN
0x1f和015具体怎么解析的，请参考485地址解析
/***/CP
MDEV_API BOOL CPMDEV_CALL CPM_IOCtrl(const char* cszDevAddr, const DEV_IOCTRL& ioctrl, int nSeq = -1);

17. 人脸机固件网络更新

/***/
功　　能　设备程序更新
参　　数
cszDevAddr　设备地址
cAppData　　程序更新结构体DEV_APPUPDATE
nSeq　操作序列
说　　明　　固件更新完成后，人脸机将自动重启，启动时间约1分钟
/***/
BOOL CPM_ULUpdate(const char* cszDevAddr, const DEV_APPUPDATE& cAppData, int nSeq = -1);

18. 人脸机远程重启

/***/
功　　能 重启设备
参　　数 cszDevAddr 设备地址

说　　明
/***/
CPMDEV_API BOOL CPM_RebootDev(const char* cszDevAddr, int nSeq = -1);

19. 人脸机组播地址设定
/***/
功　　能 设置设备所在区域地址
参　　数 cszDevAddr 设备地址 ; cszRgnAddr 设备区域地址(组播地址)
　　所有人脸机出厂默认为：224.0.1.100
说　　明 建立连接后可将设备设置到某一个区域，以后可搜索该区域下设备
/***/
CPMDEV_API BOOL CPM_ULDevRegionAddr(const char* cszDevAddr,
const char* cszRgnAddr = DEV_REGION_ADDR, int nSeq = -1);

20. 1:1人脸识别，用户远程加载
/***/
功　　能　上传实时用户
参　　数
　　cszDevAddr　设备地址
　　cUser　　用户信息结构体
　　cRtLoad　　实时加载信息
　　nSeq　　操作序列号
说　　明　设备端请求客户端上传某用户时，通过此接口上传所请求用户信息，cRtLoad为设备端请求时传过来的信息，客户端禁止修改。此函数实现远程用户加载功能。
/***/
BOOL CPM_ULRealTimeUser(const char* cszDevAddr, const DEV_USER& cUser,
const DEV_USER_REAL_LOAD& cRtLoad) ;
重要说明: 当用户在人脸机上刷卡或输入工号进行1:1识别时，该用户不存在于人脸机本地，则尝试从管理端远程加载，人脸机将卡号等相关信息发送给管理端，管理端收到该请求之后(通过回调函数)，通过该函数上传用户资料。使用该功能有两个条件1.使能DEV_WORKATT:: m_BaseSet |=DEV_REALTIME_USERLOAD。2.网络连接。

21. 用户采集实时同步
重要说明:当用户采集完成后(例如:临时用户)，在网络连接的情况下，人脸机会实时将该用户数据发送给管理端，管理端通过回调函数接收。使用该功能需要使能使能DEV_WORKATT:: m_BaseSet |= DEV_REALTIME_USERSEND

22. 实时动态记录
重要说明: 当人脸机产生验证记录，报警记录产生时，会实时将该记录发送给管理端，通过回调。使用该功能需要使能DEV_WORKATT:: m_BaseSet |= DEV_REALTIME_RECORD。

23. 管理端远程用户采集
/***/
功　　能 远程实时采集用户特征
参　　数

cszDevAddr 设备地址
cUser 用户信息
nSeq 序列号

说 明 管理端录入用户资料时，可通过该函数将用户资料发送的人脸机，控制人脸机采集该用户的人脸。远程用户采集功能，使得直接在管理端就可以采集用户，在配合远程视频的情况下采集的可视化程度和直接在机器上采集一样。如果使能了(用户采集实时同步)。远程采集的用户也会实时的发回管理端.

```
/*************************************************************************/
CPMDEV_API BOOL CPMDEV_CALL CPM_ULRealTimeUserCap(const char* cszDevAddr, const DEV_USER& cUser, int nSeq = -1);
```

24. 管理端远程查看人脸机彩色视频

```
/*************************************************************************/
```

功 能 启动视频
参 数
cszDevAddr 设备地址
hwnd 视频显示窗口句柄
说 明

```
/*************************************************************************/
CPMDEV_API BOOL CPM_StartVideo( const char* cszDevAddr, LONG hwnd, int nSeq = -1 );

/*************************************************************************/
```

功 能 停止视频
参 数 cszDevAddr 设备地址
说 明

```
/*************************************************************************/
CPMDEV_API BOOL CPM_StopVideo( const char* cszDevAddr, int nSeq = -1 );
```

重要说明:
远程视频使得管理端可以远程查看人脸机彩色视频，使远程采集功能更加完善。由于人脸机资源有限，无法做h264压缩。视频实际是通过简单处理的YUV数据通过UDP传输的，数据量较大，所以一次显示不得超过4台人脸机视频，最好只显示一台且只有在必要的时候才显示。