聯陽半導體
ITE Tech. Inc.

ITE Video intercom

Control Server SDK document

V0.2

ITE TECH. INC.

This is a Preliminary document release. All specifications are subject to change without notice.
The material contained in this document supersedes all previous material issued for the products herein referenced. Please contact ITE Tech. Inc. for the latest document(s).

All sales are subject to ITE's Standard Terms and Conditions, a copy of which is included in the back of this document.

ITE, IT9072E/IT9076E/IT9077TE/IT9078E/IT9079TE/IT9079TE-H is a trademark of ITE Tech. Inc.
All other trademarks are claimed by their respective owners.
All specifications are subject to change without notice.

Additional copies of this manual or other ITE literature may be obtained from:

ITE Tech. Inc. Tel: 886-2-29126889
Marketing Department Fax: 886-2-2910-2551, 886-2-2910-2552
7F, No.233-1, Baociao Rd., Sindian City,
Taipei County 23145, Taiwan, ROC

You may also find the local sales representative nearest you on the ITE web site.

To find out more about ITE, visit our World Wide Web at:
http://www.ite.com.tw

Or e-mail itesupport@ite.com.tw for more product information/services

修訂記錄

修訂日期	**修訂說明**	頁次
2014/10/23	初建版本 V0.1	
2014/12/31	更新版本 V0.2	

1. 1前言

本 Control Server 系统是为了配合嵌入式系统架构开发的，Control Server 可有效的管理门口机、室内机、管理机、小门口机设备以及手机 App，使上述的这些设备之间友好的协作，完成整个系统框架下的任务。

控制服务器（Control Server）是为了配合整个可视对讲系统开发的一套 PC 软件。再功能上，可视对讲/监视、门禁管理是两项主要的应用，它同时也有管理系统有对各个终端设备的交互管理，如：升级管理，设备管理，本地系统管理。同时也可提供更多的社区服务模块（社区信息发布，社区管理，三表抄送等）。本文档主要介绍整个系统的框架，以及各个模块的 SDK 说明，旨在让各开发者快速了解系统，迅速开发。

1.1 编写目的

本文档的编写为下阶段的设计、开发提供依据，为项目组成员对需求的详尽理解，以及在开发开发过程中的协同工作提供强有力的保证。同时本文档也作为项目测试评审验收的依据之一。

1.2 本产品涉及用户与角色

外来访客：例如快递员

管理员：社区工作人员，负责管理和维护整套社区系统。

住户：室内机终端和门铃机终端的使用者

1.3 文档名词约定

门口机：通常指单元门口机，一栋房子的入口处安防的机器。也是文档中指定的大门口机。

室内机：住户家中的设备终端。

小门口机：也称为别墅机或二次确认机。

管理中心机：类似于室内机的管理终端，通常是管理人员用来接听和拨出使用。用于日常事务处理。

管理中心 sever：数据存储和管理，社区管理的核心设备。同时也担任管理中心机的职责：呼叫，监视，应答。

名称换算：

大门口机/门口机 --> 大厅机 (Lobby Phone)

小门口机/室外机/门前机/门外机/别墅机/二次确认机 --> 门铃机 (Door Camera)

室内机 --> 室内机 (Indoor Phone)

管理机/管理中心机--> 管理机 (Administrator Unit)

PC 管理中心/管理中心 sever --> 控制服务器 (Control Server)

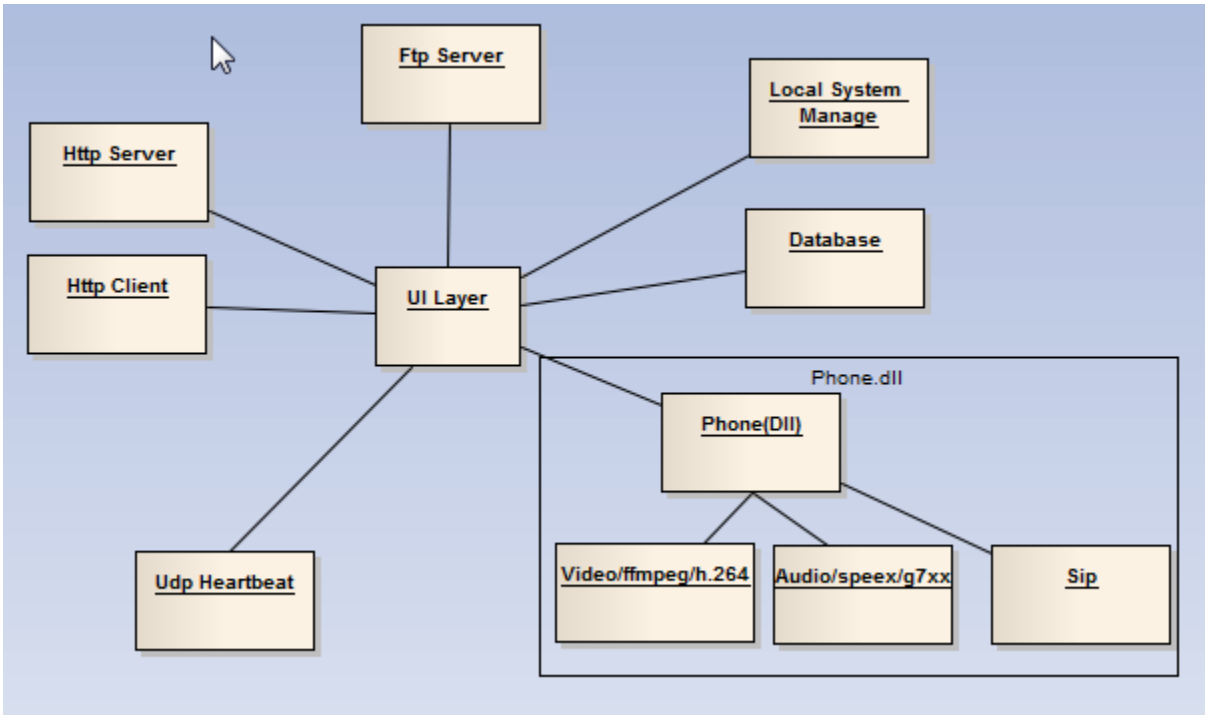
1.4 Control Server SDK 开发环境

系统：Windows 7 操作系统（双核 4G 以上）/windows xp 操作系统 带有声卡

工具：Microsoft Visual Studio 2010 开发工具

数据库支持：MySQL 关系型数据库管理系统

1.5 SDK 架构



Sdk 架构图

Http Server：Http Server 服务器端，处理客户端请求命令，并向逻辑层接口返回收到的 http 请求消息。用户只需要在回调接口中处理相应的事件请求。

Http Clinet: 封装了控制服务器所有的客户端命令。调用里面接口，可完成请求命令。

Ftp Server：控制服务器构建的 Ftp 服务，提供终端机升级服务。

UDP Heartbeat Service:心跳包，控制服务器主动下发 UDP 心跳包，检测设备在线状态。

Phone(dll): 封装的一个模块主要处理呼叫、响铃、视频/音频接受、播放等接口。支持 Audio/speex/g711 解码,Video/ffmpeg/H.264 解码，底层协商会话采用标准的 SIP 协议。

Database： 提供数据库操作接口

Local system Manager:本地设置接口。

UI Layer: UI 层，组织了所有的功能逻辑。

1.6 SDK 开发介绍

1. UI 层初始化 Control Server

ICMServer 构建的是 vs 单文档的工程，应用程序 ICMServerApp.cpp 档案的入口 CICMServerApp::InitInstance()中进行系统初始化工作。

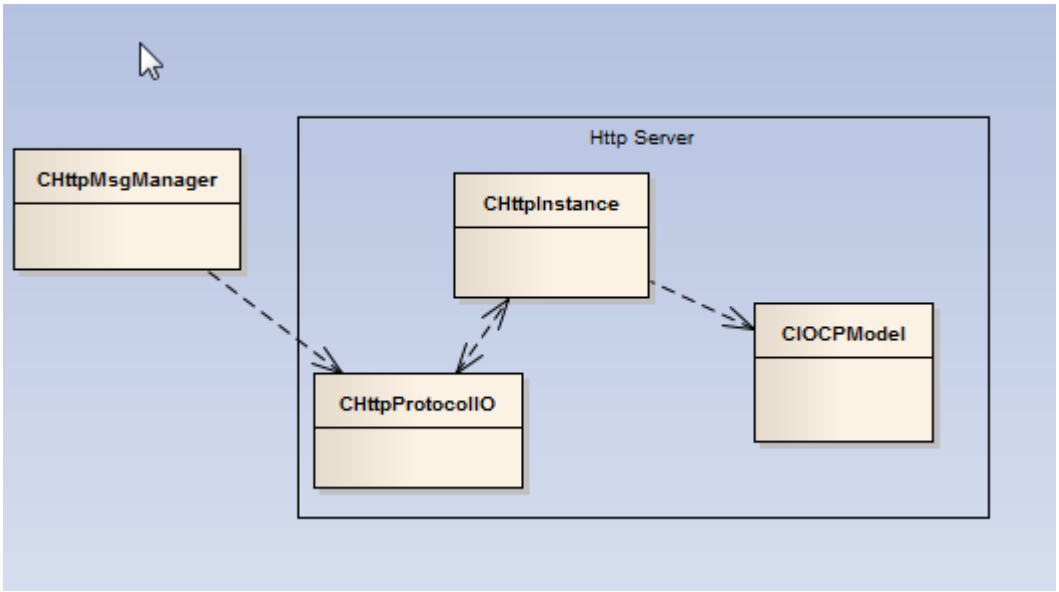
具体初始化如下所示：

```
/*获取当前系统版本*/
m_nOSVersion = GetOSDisplayString(m_szOS);
/*获取配置文件中的语言*/
CConfigFile cfg;
OnLanguage(cfg.GetSystemLanguage());
......
/*初始化并连接数据库*/
if(!ConnectDB())
{
    AfxMessageBox("database connect failed!");
}
 /*进入登录程序*/
 CDlgLogin DlgLogin;
 if(DlgLogin.DoModal() != 1)
 {
     return false;
 }
```

```
/*启动服务*/
/*start ntp server */
NtpServerBegin();
/*启动其他任务*/
InitTaskManager();
/*启动 http 服务*/
HttpServerBegin();
/*启动天气预报服务*/
WeatherServiceBegin();
```

2．Http Server

该模块主要类结构设计：



CHttpProtocolIO 主要负责解析所有的 Http 协议，同时提供接口返回 Http Response。

CHttpInstance 为 Http 服务管理类，里面可追踪所有的 http 连接。

CIOCPModel IOCP 构建 Http 服务底层类，所有的 Http 异步处理在这个类中实现（一般不需要去修改此类）。

该模块主要提供 http 服务，自定义 http 协议详见 Protocol 文档主要定义。代码在 HttpProtocolIO.h 和

HttpProtocolIO.cpp、AppInstance.h 和 AppInstance.cpp， AppInstance 是对 HttpProtocolIO 的管理。

HttpProtocolIO.h 中代码接口如下：

1） 回调函数定义

在模块中定义了一个全局的回调函数为：

```
extern ns_kenny::tentrust<stHttpMsg, void*, bool>    HttpMsgFun;
```

tentrust 模版类实现具体见 entruct.h 文件。

绑定回调函数：HttpMsgFun.plug(this, &CHttpMsgManager::CallbackServiceInterface);

2） 主要接口

功能：Http Server 向请求端 发送 Response 消息。

函数名称： SendHttpResponse

参数：stHttpMsg :包装的 http 消息结构体

```
typedef struct ST_HTTPMSG
{  int nMsg;  //消息类型
       string sip;//http请求ip
       CNetTermBase * pNet; // 请求CNetTermBase 指针，使用者不需要关心
       void * param1;      //附加指针1
       void * param2;      //附加指针2
}stHttpMsg;
```

Char * buffer:返回response字符串口

Int len :字符串长度

返回值：bool 发送成功与否。

```
*/
static bool SendHttpResponse(stHttpMsg msg,char * buffer,int len);

/*
功能：获取 Response Heade 字符串
函数名称：GetResponseHead
参数：int bodysize: Response body 长度
返回值：string 类型的字符
*/
static string GetResponseHead(int bodysize);
/
```

功能：获取 http Server 端的 ip

函数名称：GetHttpServerIp

参数：NULL

返回值：string 类型的 ip 地址

***/**

static string GetHttpServerIp();

如要添加新的协议可在 HttpProtocolIO.cpp 中可在以下三个函数中去修改。

Int GetHttpType(string s)

该函数将 http Request 字段映射为 int 型的 id 消息类型，以后判断消息用 id 来区分。如有新的消息可在此中添加消息 id 和映射。

void PostCommand(stHttpMsg msg,stHttp_Url rq);

void GetCommand(stHttpMsg msg,stHttp_Url rq);

上面两个函数是处理 Post 和 Get 的 http 请求的，如要将有效内容返回给回调函数，就在此处添加消息处理程序。

如下为 AppInstance.h 类文件中三个主要接口接口：

/*

功能：创建 Http 服务，创建一个 CHttpInstance 对象，且有且仅有一个，当再次调用的时候，返回的是已经存在的对象指针。

函数名：CreateHttpServer

参数：int nPort :http server 端口

返回值：CHttpInstance 对象指针

***/**

static CHttpInstance * CreateHttpServer(int nPort =0);//创建 Http 服务 调用此函数，创建 CHttpProtocolIO 对象。

/*

功能：初始 Http Server 信息并启动 server

函数名：InitionlizeServer

参数：NULL

返回值：**void**

*/

void InitionlizeServer();

/*

功能：停止 **Http Server**

函数名：**DeInitializeServer**

参数：**NULL**

返回值：**void**

*/

void DeInitializeServer();

详细说明:调用 CreateHttpServer 创建一个 Http Server 服务对象,InitionlizeServer 初始化并启动 Http 服务， 具体调用就在 CHttpProtocolIO 中去实现。当结束服务的时候，调用 **DeInitializeServer** 函数，即可释放所有服务资源。

3．Http Client

1) http 客户端采用了 libcurl 库，具体的接口使用请参考 libcurl 官网。
2) 该模块中主要封装了 http 客户端请求，HttpClientIO.h 主要代码接口如下所示（详细实现请查看 HttpClientIO.cpp 文件）：

```
bool SendHttpNewMessage(string ClientIp,int nCount, int nType);
bool SendOpenLock(string ClientIp, string ro);
bool SendNewCardMessage(string ClientIp, string sVer);
bool SendRebootMessage(string ClientIp);
bool SendUpdateAddrBookMsg(string ClientIp, string sFtp);
bool SendUpdateResMsg(string ClientIp, string sFtp);
bool SendSwUpgradeMsg(string ip, string sVer);
bool SendCancelSecurityMsg(string ClientIp,string ro);
bool GetDeviceCfgInfo(string ClientIp,string & sInfo);
string GetHttpResponse();

bool GetImgFormServer(string sUrl, string sImgAddr);
```

3） 接口说明：

void SendHttpMsg(string ip,char * pBuffer, int nType);

/*发送 http 提示消息*/

bool SendHttpNewMessage(string ClientIp,int nCount, int nType);

/*发送开锁消息* /

bool SendOpenLock(string ClientIp, string ro);

/*发送门禁卡更新消息*/

bool SendNewCardMessage(string ClientIp, string sVer);

/*发送重启消息*/

bool SendRebootMessage(string ClientIp);

/*发送更细地址薄消息*/

bool SendUpdateAddrBookMsg(string ClientIp, string sFtp);

/*发送更新资源文件消息*/

bool SendUpdateResMsg(string ClientIp, string sFtp);

/*发送撤销安防报警消息*/

bool SendCancelSecurityMsg(string ClientIp);

/*向 device 获取 cfg 文件*/

bool GetDeviceCfgInfo(string ClientIp,string & sInfo);

/*获取当前响应消息*/

string GetHttpResponse();

/*从服务器下载图片*/

bool GetImgFormServer(string sUrl, string sImgAddr);

4．Ftp Server

1) 在 TCP/IP 网络中，客户机可通过文件传输协议 FTP(File Transport Protocol)下载或加载文件服务器上的文件，以实现资源共享，各个终端设备 FTP Server 相连接，可访问服务器上的大量程序和信息。FTP Server 已成为互联网上的一种重要资源。在本控制服务器中，FTP Server 主要是放置升级软件文件，和各种 UPG 文件，终端机可通过 ftp url 来访问服务器，获取相应的资源。见“FtpserverIO.h”头文件：

public:

//回調函數指針

static pFtpCallBackFun mpOnBeginFun;

static pFtpCallBackFun mpOnProgressFun;

```
        static pFtpCallBackFun mpOnEndFun;
        static string m_strCurDir;
    public:/*向外提供接口*/
        bool RunFtpServer();
        bool StopFtpServer();
        void SetCurrentDir(string sDir);

        //注冊函數
        void RegBeginFun(pFtpCallBackFun pFun);
        void RegProgressFun(pFtpCallBackFun pFun);
        void RegEndFun(pFtpCallBackFun pFun);
```

2) 接口说明

```
/*启动 ftp service*/
bool RunFtpServer();
/*停止 ftp Service*/
 bool StopFtpServer();
/*设置 ftp 根目录*/
 void SetCurrentDir(string sDir);

 //注冊函數
 /*注冊升级开始回调消息函数*/
void RegBeginFun(pFtpCallBackFun pFun);
 void RegProgressFun(pFtpCallBackFun pFun);
/*注冊升级结束回调消息函数*   /
 void RegEndFun(pFtpCallBackFun pFun);
```

详细说明：

在使用此服务的时候，需要 RunFtpServer() 函数启动 ftp service ，然后 SetCurrentDir(string sDir) 设置 ftp 根目录，如果调用设置，ftp service 会默认为当前程序的执行目录，当程序结束的时候调用 StopFtpServer()，服务将被终止。

5．Database

每个数据库表类都有一个基类 CADODatabase，应用到的数据表类都是有基类派生出来的，CADORecordset 是 Ado 操作产生的数据集接口包装，详细见代码。

数据库表设计：（以下数据字段类型都是参考 mysql 数据支持类型，如果使用其他关系型或者非关系型数据库，请自行转换）

1、数据表：设备表（device)

字段	_id	_ip	_roomid	_alias	_mac	_status	_type	_sm	_gw	_aVer	_cVer
描述	PRIMARY KEY	设备 ip	设备地址	设备别名	设备 mac 地址	在线状态	设备类型	子网掩码	网关	设备地址薄版本号	设备门禁卡版本号
类型	INTEGER	Varchar(20)	Varchar(20)	varchar(50)	varchar(50)	INTEGER	INTEGER	varchar(50)	varchar(50)	varchar(50)	varchar(50)
其他	NOT NULL AUTO_INCREMENT	NOT NULL									此字段在门口机上才有效

设备类型（定义了 7 种设备类型，可以自行在此基础上增加）：

0：管理中心。1：小门口机。2：单元门口机。3：栋门口机。4：小区门口机（围墙机）5：室内机；6 管理中心机

2、数据表：留影留言记录（leaveword)

字段	_id	_filenames	_src_addr	_dst_addr	_ttime	_readflag
描述	Primary key	留影留言文件路径	Lobby phone address	Indoor Address(无分机）	留影留言时间点	读标记
类型	INTEGER	Varchar(250)	Varchar(20)	Varchar(20)	Varchar(20)	Varchar(20)
其他	NOT NULL AUTO_INCREMENT					

3、数据表：门禁卡信息（iccard）

字段	_id	_icno	_roomid	_username	_ictype	_icpassword	_available	_time	_work_begin	_work_end
描述	Primary key	Ic 卡序列号	开卡地址	开卡人	开卡类型		有效	开卡时间	有效时段其实时间	有效时段结束时间
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)
其他										

4、数据表：门禁卡可开门口机表（icmap)

字段	_id	_icno	_entrancedoor
描述	Primary key	门禁卡号	门口机地址
类型	INTEGER	VARCHAR(20)	VARCHAR(20)
其他		NOT NULL	NOT NULL

5、数据表：安防事件表（EventWarn)

字段	_id	_src_addr	_ttime	_handlestatus	_handletime	_type	_channel	_action	_handler
描述	Primary key	上报地址	上报时间	处理状态	处理事件时间	事件类型	事件通道	动作	处理人
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	INTEGER	VARCHAR(20)	INTEGER	INTEGER	VARCHAR(20)	VARCHAR(20)
其他									

6、数据表：对讲记录上报（EventCallout)

字段	_id	_from_addr	_to_addr	_ttime	_type	_img_addr	
描述	Primary key	主叫地址	被叫地址	记录时间点	记录类型	记录影像图片	
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)	INTEGER	VARCHAR(255)	
其他							

7、数据表：普通事件表（EventCommon)

字段	_id	_src_addr	_ttime	_handlestatus	_handletime	_type	_content	_action	_handler
描述	Primary key	事件上报地址	事件上报时间	处理状态	处理时间	事件类型	上报内容	动作	处理人
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	INTEGER	VARCHAR(20)	INTEGER	VARCHAR(255)	VARCHAR(20)	VARCHAR(255)
其他									

8、数据表：升级文件表（Upgrade)

字段	_id	_filepath	_ver	_type	_time
描述	Primary key	升级文件保存地址	升级文件版本号	升级文件类型	保存时间
类型	INTEGER	VARCHAR(255)	VARCHAR(20)	INTEGER	VARCHAR(20)
其他					

9、数据表：发布信息（publishinfo)

字段	_id	_topic	_dst_addr	_ttime	_type	_fmt	_readflag	
描述	Primary key	发布主题	发布目的地址	发布时间点	发布信息类型	格式	读标记	
类型	INTEGER	VARCHAR (250)	VARCHAR (20)	VARCHAR (20)	VARCHAR (20)	VARCHAR (20)	INTEGER	
其他								

10、数据表：住户信息表（HolderInfo)

字段	_id	_name	_sex	_phoneno	_roomid	_isholder	
描述	Primary key	住户姓名	住户性别	住户手机号	住户房号	是否房主	
类型	INTEGER	VARCHAR(20)	INTEGER	VARCHAR(20)	VARCHAR(20)	INTEGER	
其他							

11、数据表：楼栋属性表（buildproperty)

字段	_qu	_dong	_type		
描述	区	栋	类型		
类型	VARCHAR(20)	VARCHAR(20)	INTEGER		
其他	Primary key	Primary key			

说明：类型（三种类型）：独栋无单元，多单元型，别墅型

12、数据表：门口机密码表（doorbellpassword)

字段	_id	_roomid	_password	_ttime			

描述	Primary key	门口机地址	门口机密码	上传时间点			
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	VARCHAR(20)			
其他							

13、数据表：系统用户表（user)

字段	_id	_userno	_usename	_powerid	_password
描述	Primary key	用户 id	用户名称	权限 id	用户登录密码
类型	INTEGER	VARCHAR(20)	VARCHAR(20)	INTEGER	VARCHAR(20)
其他					

14、数据表：用户权限表（authority)

字段	_id	_name	_authority
描述	Primary key	权限名称	权限
类型	INTEGER	VARCHAR(20)	INTEGER
其他			

15、数据表：地址薄规则表（AddrAssociate）

字段	_id	_addrA	_typeA	_addrB	_typeB	_des
描述	Primary key	源地址	规则类型	目标地址	目标地址设备类型	描述

类型	INTEGER	VARCHAR(20)	INTEGER	VARCHAR(20)	INTEGER	VARCHAR(20)
其他						

16、数据表：心跳包日志记录（heartbeatlog）

字段	_id	_log	_logtime
描述	Primary key	记录内容	记录产生时间
类型	INTEGER	VARCHAR(255)	DATETIME
其他		NOT NULL	

上述数据库基本操作接口都整理在 database_interface.h 中，这部分操作接口可见源码，接口如下如下所示：

```
/*user table*/
extern bool db_InsertUser(stUserInfo user);
extern bool db_GetUserByUserName(stUserInfo &user,std::string sUserName);

/*authrity table*/
extern int  db_GetAllAuthrityInfomations(vector<stAuthority> &va);
extern bool db_RemoveAuthrityRecord();

/*end*/
/*device operator*/
extern int  db_GetAllDevices(std::vector<stDevice> &vcDev,string where = "");
extern bool  db_GetDeviceByIp(stDevice & dev, string ip);
extern int  db_GetDeviceByAddress(std::vector<stDevice> &vcDev, string roomid);
extern bool db_GetDeviceByMac(stDevice &dev, string mac);
extern int  db_UpdateDeviceStatus(string ip, bool bOnline = true);
extern bool  db_UpdateDeviceMacAddress(string ip, string mac);
```

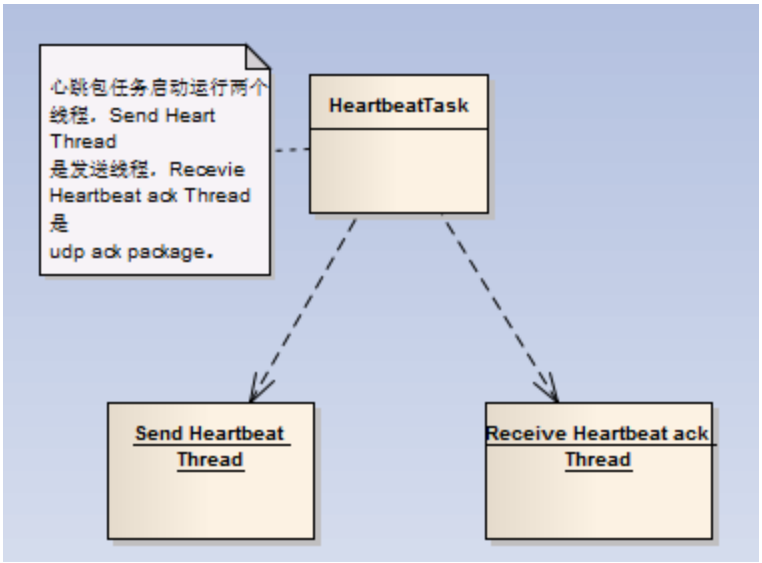
```
extern bool db_UpdateDeviceAddressbookVer(string address, string sVer);
extern bool db_InsertDevice(stDevice dev);
extern bool db_RemoveDeviceByIp(string ip);
extern bool db_RemoveAllDevices();
extern bool  db_GetDeviceAddressbookVer(string address,string &sVer) ;
extern bool  db_UpdateDeviceVerInfo(string ip, char * paVer, char * pcVer, char * pfVer);
/*leave word operator*/
extern int  db_GetLwRecords(std::vector<stLeaveWord> &ll, std::string where = "");
extern bool db_InsertLwRecord(stLeaveWord lw);
extern bool db_UpdateLwStatusByid( int id,bool bRead = true);
extern bool db_RemoveLwRecordByid( int id);
extern bool db_RemoveLwRecordByTime(string stime);
/*doorbell card*/
extern int  db_GetCardRecords(vector<stICinfo> &vi, string where = "");
extern int  db_GetCardRecords(vector<stICmap> &vi, string where="");
extern int  db_GetAllCardByAddress(vector<stICmap> &vi,string address);
extern int  db_GetAllDeviceAddressByIcno(vector<stICmap> & vi,string icno);
extern bool db_InsertIcCard(stICinfo ic);
extern bool db_UpdataIcStatusIcno(string icno, bool bValid = true);
extern bool db_RemoveCardRecordByno(string icno);

/*event  function*/
extern bool db_InsertWarnEventRecord(stEvent_Warn ew);
extern bool db_InsertCommonEventRecord(stEvent_Common ec);
extern bool db_InsertCalloutEventRecord(stEvent_Callout ec);
extern bool db_InsertOpenDoorEventRecord(stEventOpenDoor eod);

extern int  db_GetAllWarnEventRecord(std::vector<stEvent_Warn> &vew);
extern int  db_GetAllCommonEventRecord(std::vector<stEvent_Common> &vew);
extern int  db_GetAllCalloutEventRecord(std::vector<stEvent_Callout> &vew);
extern int  db_GetAllOpenDoorEventRecord(std::vector<stEventOpenDoor> &veo);

extern bool db_RemoveWarnEventRecordByTime(std::string sTime);
```

```
extern bool db_RemoveCommonEventRecordByTime(std::string sTime);
extern bool db_RemoveCalloutEventRecordByTime(std::string sTime);
extern bool db_RemoveOpenDoorEventRecordByTime(std::string sTime);
/*password manager function*/
extern int  db_GetDoorPasswordRecords(std::vector<stDoorbellPassword> & vdp,string where ="");
extern bool db_GetDoorPasswordByAddress(string & sPswd, string address);

/*upgrade record function*/
extern int  db_GetAllUpgradeRecord(vector<stUpdateFile>& vuf);
extern bool db_GetUpgradeRecordByTime(stUpdateFile &uf,string sTime);
extern bool db_RemoveUpgradeRecordByTime(string sTime);
extern bool db_InsertUpgradeRecord(stUpdateFile uf);

/*publish information function*/
extern int  db_GetPublishInformation(vector<stPublishInfo> &vpi,string where="");
extern bool db_InsertPublishInformation(stPublishInfo pi);
extern bool db_RemovePublishInformationByTime(string stime);
//extern

/*heartbeatlog functions*/
extern int  db_GetHeartbeatLog(vector< stHeartbeatLog> & vhl);
extern bool db_InsertHeartbeatLog(stHeartbeatLog hl);
/*end*/

extern bool db_InsertPhotographRecord(stPhotograph sp);
extern bool db_GetPhotographRecordByAddressAndTime(stPhotograph &sp, string saddress, string stime);


/*build Property*/
extern bool db_InsertBuildPropertyRecord(stBuildProperty bp);
extern bool db_RemoveBuildPropertyRecordByBuild(stBuildProperty bp);
extern int  db_GetAllBuildPropertyRecord(vector<stBuildProperty> &vc);
```

上述提供了基本的操作接口，如果要增加其他接口，可查看 ado.h 中的函数接口进行自定义接口。

6．UDP Heartbeat

该功能是一个任务，主要来检测设备在线状态。具体代码见工程中 HeartBeatTask.h 和 HeartBeatTask.cpp 文件中，使用 udp 端口为 49201。



任务架构

规则：

3s 钟发送一个检测包，收到回复了就停止发送，最多发送 3 次，如果三次都没收到终端设备的回复，那么就认定该设备已经离线。

代码见“HeartbeatTask.h”中：

```
#pragma once
#include <vector>
#include <list>
#include <map>
using namespace std;

#define UDP_HEARTBEAT_ON_LINE    1    /*on line*/
#define UDP_HEARTBEAT_OFF_LINE 0    /*off line*/
class CHeartBitTask
{
public:
    CHeartBitTask(void);
```

```
    ~CHeartBitTask(void);
public:
    typedef struct _HeartBitInfo
    {
        string ip;
        int count;

    }HeartBitInfo;

    static DWORD __stdcall ThreadSendProc(LPVOID lparam);
    static DWORD __stdcall ThreadRecvProc(LPVOID lparam);

    list<HeartBitInfo> m_vSendList;
    map<string,HeartBitInfo> m_vRecvList;
    list<HeartBitInfo> m_vAllList;

    bool m_bSendAuto;
    bool m_bRecvAuto;
    SOCKET local;

private:
    HANDLE m_hSendThread;
    HANDLE m_hRecvThread;



    typedef void (__stdcall *CallBackFunc)(void *,int *,char *);
    CallBackFunc m_pCallBackFunc;
    void *m_Object;
public:
    int    InitUdpSocket(int nPort);


public:
template <class ClassName> void RegRecvFunc(void *pObject, void (__stdcall ClassName::*pFunc)(int ,char
*));
```

```
    /*初始化任務發送 List*/
    void InitDeviceList(vector<string> vip_address);
    void BeginTask();
    void EndTask();
};
```

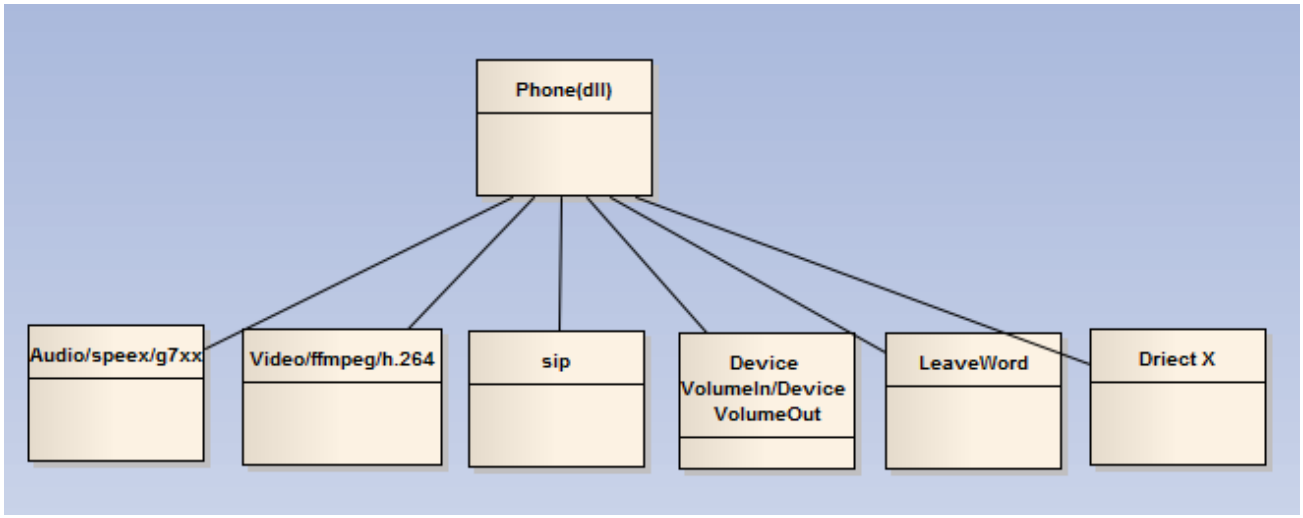
用户使用的时候只需要调用一下 4 个接口即可，如下：

```
/*
功能：模版函数，注册回调函数，返回设备在线消息
函数名：RegRecvFunc
参数：void * pObject :  回调函数对象指针
        void (__stdcall ClassName::*pFunc)(int ,char *):回调函数
返回值：void
说明：只有调用此函数注册了，才能返回设备状态消息
*/
template <class ClassName> void RegRecvFunc(void *pObject, void (__stdcall ClassName::*pFunc)(int ,char *));
/*
功能：初始化任務發送 List
函数名：InitDeviceList
参数：vector<string> vip_address:设备 ip list
返回值：void
说明：
*/
    void InitDeviceList(vector<string> vip_address);
/*启动任务*/
    void BeginTask();
/*结束任务*/
    void EndTask();
```

备注：启动任务前要先 RegRecvFunc 注册回调函数，然后 BeginTask 启动任务，并 InitDeviceList 初始化系统设备表，当结束任务时候，请调用 EndTask 释放所有任务资源。

7．Phone (Phone.dll)

该层模块集成了 Video /Audio 解码部分、Sip 通信部分、Direct X、留影留言处理 Module 、Device Volume In/Device Volume Out 如下图所示：



模块图（Phone.dll)

a、Video/Audio 采用最新的 fmpeg 库，audio 引入了 Speex 和 g7xx 库。

b、直接调用 Phone.dll 接口可以完成基本的 Sip 动作：sip 呼叫，sip 呼叫停止、sip 监视，sip 停止监视

c、在 Phone.dll 中已经建立了一个视频 Show Task，sip 处理->视频接受->视频显示都是在 dll 中去处理，只需要在调用初始化 dll 的时候，将显示视频的窗口的句柄（hwnd 或者 Cwnd *)传入即可。

接口见 MMDataType.h 和 iphone_dll_interface.h 中。

Phone.dll 接口说明：

```
/*
功能：初始化对讲功能
函数名：dll_init
参数：dll_phoneCallBack  func  回调函数
返回：0：初始化成功  -1:初始化失败
说明：所有的消息都是通过回调函数返回， 不要在回调函数里面添加阻塞事件。
typedef  void (WINAPI * dll_phoneCallBack)(sIteDataParam  d);
```

结构体定义如下：

```
typedef struct sIteDataParam{
        BSTR address;
        BSTR ip;
        int  msgtype;
    }sIteDataParam;
```

返回消息消息包括：

```
/*meet operator */
emMEET_INVAL      = 0,                  //标志无效
        emMEET_REQUEST    = 1,      //请求对讲
        emMEET_ACCEPT     = 2,      //接受对讲
        emMEET_REFUSE     = 3,      //拒绝对讲
        emMEET_TIMEOVER   = 4,      //请求超时
        emMEET_STOP       = 5,      //挂断对讲
        emMEET_OPERA      = 6,      //辅助操作
        emMEET_FAILED     = 7,      //操作失败
        /*watch operator*/
        emWATCH_REQUEST   = 8,      //请求监视
        emWATCH_BUSY      = 9,      //监视正忙
        emWATCH_TIMEOVER  = 10,     //监视超时
        emWATCH_STOP      = 11,     //停止监视
        emWATCH_ACCEPT    = 12,     //接受监视

        /*leave word operator*/

        emLW_STOP           = 14,
        emLW_REQUEST        = 15,
        emLW_DOWNLOAD_REQUEST = 16,
        /*Phone Records upload*/
        emPHONERECORDS_FROM = 17,
        emPHONERECORDS_TO   = 18
```

```
    } TMeetDataFlag;
*/
extern "C" __declspec(dllexport) int __stdcall dll_init(dll_phoneCallBack   func)
```

Sip 动作提供了一个 dll_Dataout 接口，flag 参数包含了 sip 所有的动作，如下所见。

```
/*
功能：提供 sip 呼叫接口
函数名：dll_Dataout
参数：int nCallType:消息类型
      char * ip     :操作设备 ip
      char * address：操作设备地址
返回值：true or false
说明：nCallType 操作类型：
                          = emMEET_REQUEST          //请求对讲
                          = emMEET_ACCEPT       //接受对讲
                          = emMEET_REFUSE       //拒绝对讲
                          = emMEET_STOP         //挂断对讲
                          = emWATCH_REQUEST     //请求监视
                          = emWATCH_STOP        //停止监视

    传参只需要关注上面结构体字段即可。
*/
extern "C" __declspec(dllexport) bool __stdcall dll_DataOut(int nCallType, char * ip, char * address
)


/*
void dll_Handle(HWND); 绑定视频输出窗口的句柄

//音量控制操作函数
/*
功  能：设置麦克风声音
函数名：dll_SetVolumeImport
参数类型: DWORD  输入音量值
返回值：void
*/
```

```
void dll_SetVolumeImport(DWORD);
/*
功能：设置输出音量
函数名：dll_SetVolumeExport
参数类型：DWORD   输出音量值
返回值：void
*/
void dll_SetVolumeExport(DWORD);
/*
功能：获取当前输入音量的值
函数名：dll_GetVolumeImport
参数类型：void
返回值：DWORD  返回当前输入音量值
*/
DWORD ll_GetVolumeImport();
/*
功能：获取当前输出音量值
函数名：dll_GetVolumeExport
参数：void
返回值：DWORD  返回当前输出音量值
*/
DWORD dll_GetVolumeExport();

/*
功能：获取当前对讲的通话状态
函数名：dll_CallState
参数：NULL
返回值：nsMM::TMeetStateFlag      当前通话状态的值
说明:typedef enum {
            emMEET_STATE_FREE,              //空闲状态
            emMEET_STATE_REQUESTING,        //正在呼叫
            emMEET_STATE_CALLING,           //正在通话
      } TMeetStateFlag;
*/
extern "C" __declspec(dllexport) ITE::TMeetStateFlag __stdcall dll_CallState()

/*
```

功能：获取当前的通话类型

函数名：dll_CallType

参数：NULL

返回值：nsMM::TCallType　呼叫类型

说明：typedef enum {

emNOCALL,//无通话状态

emTOCALL,//被呼叫状态

emFORCALL,//主动呼叫状态

emWATCH //监视状态

}　TCallType;

*/

extern "C" __declspec(dllexport) ITE::TPhone::TCallType __stdcall dll_CallType()

/*

功能：监视状态返回

函数名：dll_WatchState

参数：NULL

返回值：TWatchStateFlag　监视状态

说明：　typedef enum {

emWATCH_STATE_FREE,　　//空闲状态

emWATCH_STATE_BUSY,　　//正在监视

} TWatchStateFlag;ype;

*/

extern "C" __declspec(dllexport) ITE::TWatchStateFlag　__stdcall dll_WatchState()

/*play record file*/

- 初始化 Phone.dll

  使用该 dll 的时候要调用初始化接口，sdk 中见 DlgVideoIntercom.cpp 的 CDlgVideoIntercom::OnInitDialog()接口中

```
/*初始化 dll 和传入回调函数指针，所有 Phone.dll 产生的消息将在 DllCallBack 中收到*/
dll_init(DllCallBack);
/*初始化视频窗口句柄*/
dll_Handle(GetDlgItem(IDC_PLAYBOX));
```

➢ 呼叫/监视接口

Sdk 中见 DlgVideoIntercom.cpp 的 CDlgVideoIntercom::DoRequest(string _address, bool watch,bool ipcall)中

```
/*调用 dll 接口发起 sip 呼叫/监视终端设备*/
dll_DataOut((int)flag, (char *)device.ip.c_str(), (char *)device.addr.c_str());
```

➢ 释放 dll

sdk 中见 DlgVideoIntercom.cpp 的 CDlgVideoIntercom::~CDlgVideoIntercom()析构函数中

```
/*释放 dll 中所有的资源*/
dll_uninit();
```

8．Local System Manage

用户登录详见 SDlgLogin.h 和 DlgLogin.cpp 中。

用户数据库备份详见 DlgDbBackup.h 和 DlgDbBackup.cpp 中。

系统参数设置详见 DlgSystemSetting.h 和 DlgSystemSetting.cpp

9．其他开发代码说明

地址薄生成 xml 源码见 AddrBookOperator.h 和 AddrBookOperator.cpp 中。

升级资源打包（UPG/PKG)接口见 pkg_tool_code.h 和 pkg_tool_code.cpp 中。