```
#include "../SMS_Ret.h"
#include "../ec_config.h"
#include "deviceimb_proc.h"
#include "../sms_timeconvert.h"
#include "../systemconfig/SMS_SystemConfig.h"
#include "../serverstate/SMS_ServerState.h"
#if defined(_WIN32)
#include "../FileManager.h"
#else
#include "../FileManager_linux.h"
#endif


#define MM_LINUX
#define MVC2API_NETWORK_ONLY

/// 是否真正使用用设备播放，用于放映模拟
#define CHECK_DEVICE_MVC        1

/// 是否调用清除设备缓存函数
#define CLEAR_CACHE     0

/// 每帧视频的最大空间
#define MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER             2097152         /// 2 *
1024 * 1024
/// 每帧音频的最大空间
#define MAX_LENGTH_AUDIOFRAME_OUTPUT_BUFFER             360000          ///

/// 开始播放前，预先下发到设备中的音视频帧数，用于缓存
#define CACHE_FRAME_COUNT                                               20

#define TEST_CPL_LOOP                                                   0

#define MAX_RUNNING_DELAY_TIME                                  3000



extern char g_sms_configfile[255];
extern Sms_SystemConfig g_systemconfig;
extern Sms_ServerState g_serverstate;

#if defined(_WIN32)
void play_thread(LPVOID pvoid)
#else
void *play_thread(LPVOID pvoid)
#endif
{
        sms_timeconvert timeconvert;
        char currtime_str[50];
        time_t time_start = 0;
        time_t time_end = 0;

#if 1
        time_start = time(&time_start);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("play procedure start:%s\n",currtime_str);
#endif

        deviceimb_proc *pproc = (deviceimb_proc *)pvoid;
        int loopcount = 1;
```

```
#if TEST_CPL_LOOP
        loopcount = 100;
#endif

        for(int i = 0; i < loopcount; i++)
        {

#if TEST_CPL_LOOP
                printf("loop count:%d\n",loopcount);
                pproc->testcplloop(16,0);
#endif

                pproc->play_procedure();

        }

        g_serverstate.Set3dMode(0);

#if 1
        time_end = time(&time_end);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("play procedure exit:%s\n",currtime_str);
        printf("play procedure seconds:%d\n",time_end-time_start);
#endif

#ifndef _WIN32
        //pthread_detach(pthread_self());

        pthread_exit(0);
#endif


}

#if defined(_WIN32)
void time_thread(LPVOID pvoid)
#else
void *time_thread(LPVOID pvoid)
#endif
{
        deviceimb_proc *pproc = (deviceimb_proc *)pvoid;

        pproc->time_procedure();

#ifndef _WIN32
        //pthread_detach(pthread_self());
        pthread_exit(0);
#endif

}
void deviceimb_proc::testcplloop(int cplid,int enterpoint)
{
        m_isplaying = 1;
        m_cplid = cplid;
        m_cplframe_index = enterpoint;
}
deviceimb_proc::deviceimb_proc()
{
        m_cplid = -1;
        m_imbid = -1;
        m_device_init_ok = 0;
```

```
        m_cached_frames_ok = 0;
        m_isloaded = 0;
        m_isplayed = 0;
        m_ismanu_schedule_mode = 0;
        m_isshift_imb_device = 0;

        m_count_cache_frame = CACHE_FRAME_COUNT;
        m_device_init_ok = 0;
        m_isplaying = 0;

        m_pimbinfo = NULL;

        m_pmvcdevice = NULL;
        m_psecuritymanager = NULL;
    m_pdecoder_j2k = NULL;
    m_pdecoder_j2k_right = NULL;
        m_pmpeg2dec = NULL;
        m_psubtitleDec = NULL;
        m_pdecoder_pcm = NULL;
        m_poutput_video = NULL;
        m_poutput_video_right = NULL;
        m_poutput_audio = NULL;
        m_pplaybackcontrol = NULL;

        reinit_playdata();

        m_ploginfo_insert = NULL;
        m_logcontrol.NewSpaceLog(&m_ploginfo_insert);

        m_isgetting_certorlog = 0;
}
deviceimb_proc::~deviceimb_proc()
{
        m_logcontrol.DeleteSpaceLog(&m_ploginfo_insert);
}
int deviceimb_proc::RegisterImbDevice(int imb_id,DEVICE_IMB_INFO
*pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;
        Sms_Cpl cpl_control;

        if( imb_id <=0 ||
                NULL == pdeviceimbinfo)
        {
                return SMS_PARAMETER_ERROR;
        }

        imbs_table.GetItemById(imb_id,pdeviceimbinfo->pimbitem);

        if(!strcmp(pdeviceimbinfo->pimbitem->id,""))
        {
                return URL_IMBCONTROLLER_NO_IMBID_ERROR;
        }

        m_imbid = imb_id;

//      if(strcmp(pdeviceimbinfo->pimbitem->cpl_id,"-1") &&
//              strcmp(pdeviceimbinfo->pimbitem->cpl_id,""))
//      {
//              cpl_control.GetCplInfoById(atoi(pdeviceimbinfo->pimbitem-
>cpl_id),pdeviceimbinfo->pcplinfo,1);
//      }
```

```
        return ret;
}
int deviceimb_proc::GetDeviceInfo(int imb_id,DEVICE_IMB_INFO *pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;
//      Sms_Cpl cpl_control;

        if( imb_id <=0 ||
                NULL == pdeviceimbinfo)
        {
                return SMS_PARAMETER_ERROR;
        }

#if 1
        printf("imb:%d\n",imb_id);
#endif

        imbs_table.GetItemById(imb_id,pdeviceimbinfo->pimbitem);

        if(!strcmp(pdeviceimbinfo->pimbitem->id,""))
        {
                return URL_IMBCONTROLLER_NO_IMBID_ERROR;
        }

//      if(strcmp(pdeviceimbinfo->pimbitem->cpl_id,"-1") &&
//              strcmp(pdeviceimbinfo->pimbitem->cpl_id,"") )
//      {
//              cpl_control.GetCplInfoById(atoi(pdeviceimbinfo->pimbitem-
>cpl_id),pdeviceimbinfo->pcplinfo,1);
//      }

        return ret;
}
int deviceimb_proc::GetDeviceInfo(DEVICE_IMB_INFO *pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;

        if(NULL == pdeviceimbinfo)
        {
                return SMS_PARAMETER_ERROR;
        }

        ret = GetDeviceInfo(m_imbid,pdeviceimbinfo);

        return ret;
}
int deviceimb_proc::UpdateDeviceInfo(DEVICE_IMB_INFO *pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;

        if(NULL == pdeviceimbinfo)
        {
                return SMS_PARAMETER_ERROR;
        }

#if 1
        printf("imb:%s\n",pdeviceimbinfo->pimbitem->id);
#endif

        ret = imbs_table.UpdateItem(pdeviceimbinfo->pimbitem);
```

```
        return ret;
}
int deviceimb_proc::GetSmReportLog(unsigned char *pbufflog,int *plen,const char
*pstart_time,const char *pend_time,char *plast_log_time,int maxlen)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;
    uint32_t buffersize=0;
        MvcDevice mvcdevice;
        SecurityManager *psecuritymanager = NULL;
        uint64_t starttime = 0;
    uint64_t endtime = 0;
    uint64_t lastlogtime = 0;
        sms_timeconvert timeconvert;
        SMS_DATETIME date;
        TIME_FORMAT_TYPE time_format = TIME_FORMAT_0;
        DEVICE_IMB_INFO *pimbinfo = NULL;
        ImbsTable imbs_table;

        NewSpaceDeviceImbInfo(&pimbinfo);
        ret = imbs_table.GetItemById(m_imbid,pimbinfo->pimbitem);
        if(SMS_SUCCESS != ret)
        {
                return ret;
        }

        if(m_isplaying)
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return URL_IMBCONTROLLER_IMB_GETSECURITYLOG_INPLAYING_ERROR;
        }

        if(m_isgetting_certorlog)
        {
                return URL_IMBCONTROLLER_DEVICE_USING_ERROR;
        }
        m_isgetting_certorlog = 1;

        ret = Connect_Test(pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        ret = get_mvcdevice(&mvcdevice,pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        ret = new_securitymanager(&mvcdevice,&psecuritymanager,pimbinfo-
>pimbitem->chain_file,pimbinfo->pimbitem->private_file);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }
```

```
#if 0
//      starttime = time(NULL) - 100000;
        starttime = 0;
   //uint64_t endTime = 1399445920;
    endtime = time(NULL);
#endif

#if 1
        timeconvert.ConvertTimeStrToInt((char *)pstart_time,(time_t
        *)&starttime,time_format);
        timeconvert.ConvertTimeStrToInt((char *)pend_time,(time_t
        *)&endtime,time_format);
#endif

        buffersize = *plen;
        mvc_ret = psecuritymanager->getLogReport(pbufflog, &buffersize,
        starttime, endtime, &lastlogtime);
        if(MMRC_Ok != mvc_ret)
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"get
                securitymanager log is error! Error code is:%d", mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                printf("get securitymanager log is error! Error code is:%d\n",
                mvc_ret);
                DeleteSpaceDeviceImbInfo(&pimbinfo);

                if(MMRC_SM_LogRangeEmpty == mvc_ret)
                {
                        m_isgetting_certorlog = 0;
                        ret = delete_securitymanager(&psecuritymanager);
                        return
                        URL_IMBCONTROLLER_IMB_GETSECURITYLOG_EMPTY_ERROR;
                }

                m_isgetting_certorlog = 0;
                ret = delete_securitymanager(&psecuritymanager);
                return URL_IMBCONTROLLER_IMB_GETSECURITYLOG_ERROR;
        }
        else
        {
//              printf("get securitymanager log is success! Success code is:
%d\n", ret);
        }

        *plen = buffersize;

        time_format = TIME_FORMAT_2;

        timeconvert.ConvertTimeIntToStr(lastlogtime,plast_log_time,maxlen,time_
        format);

        ret = delete_securitymanager(&psecuritymanager);

        DeleteSpaceDeviceImbInfo(&pimbinfo);

        m_isgetting_certorlog = 0;
```

```
        return ret;
}
int deviceimb_proc::GetSmDeviceCert(unsigned char *pbuffcert,int *plen,int
maxlen)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;
    uint32_t buffersize=0;
        MvcDevice mvcdevice;
        SecurityManager *psecuritymanager = NULL;
        DEVICE_IMB_INFO *pimbinfo = NULL;
        ImbsTable imbs_table;

        NewSpaceDeviceImbInfo(&pimbinfo);
        imbs_table.GetItemById(m_imbid,pimbinfo->pimbitem);

        *plen = 0;
        memset(pbuffcert,0,maxlen);

        if(m_isplaying)
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return URL_IMBCONTROLLER_GETCERT_INPLAYING_ERROR;
        }

        if(m_isgetting_certorlog)
        {
                return URL_IMBCONTROLLER_DEVICE_USING_ERROR;
        }
        m_isgetting_certorlog = 1;

        ret = Connect_Test(pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        ret = get_mvcdevice(&mvcdevice,pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        ret = new_securitymanager(&mvcdevice,&psecuritymanager,pimbinfo-
        >pimbitem->chain_file,pimbinfo->pimbitem->private_file);
        if(SMS_SUCCESS != ret)
        {
                m_isgetting_certorlog = 0;
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

   /// XLQ:要获取证书的类型，一共有四种，分别是：0代表设备证书，1代表根证书，2代表中间证
    书，100代表日志证书
    uint32_t which = 0;
        buffersize = maxlen;

        mvc_ret = psecuritymanager->getCertificate(which, pbuffcert,
        &buffersize);
```

```
    if(MMRC_Ok != mvc_ret)
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"get
                securitymanager cert is error! Error code is:%d", mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                m_isgetting_certorlog = 0;
                printf("get securitymanager cert is error! Error code is:%d\n",
                mvc_ret);
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                ret = delete_securitymanager(&psecuritymanager);
                return URL_IMBCONTROLLER_GETCERT_ERROR;
        }
        else
        {
//              printf("6SecurityManager is success! Success code is:%d\n",
ret);
        }

#if 0
    printf("certificate size is:\n[%d]\n", buffersize);
    printf("Certificate data is:\n%s\n", pbuffcert);
#endif

        *plen = buffersize;

        ret = delete_securitymanager(&psecuritymanager);

        DeleteSpaceDeviceImbInfo(&pimbinfo);

        m_isgetting_certorlog = 0;

        return ret;
}
int deviceimb_proc::SetScheduleMode(int ismanu_schedule_mode)
{
        int ret = SMS_SUCCESS;
        return ret;
}
int deviceimb_proc::UpdateImbBoardVersionInfo()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;
        MvcDevice mvcdevice;
        DEVICE_IMB_INFO *pimbinfo = NULL;
        ImbsTable imbs_table;
        VersionValue imbversion;
        int uid = 0;

        NewSpaceDeviceImbInfo(&pimbinfo);
        ret = imbs_table.GetItemById(m_imbid,pimbinfo->pimbitem);
        if(SMS_SUCCESS != ret)
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        ret = get_mvcdevice(&mvcdevice,pimbinfo);
        if(SMS_SUCCESS != ret)
```

```
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

#if 0
        imbversion = 0;
        imbversion = mvcdevice.getAPIVersion();
        sprintf(pimbinfo->pimbitem->api_version,"%d.%d.%d.%d",
                imbversion.getVersion(),
                imbversion.getRevision(),
                imbversion.getBuildVersion(),
                imbversion.getBuildRevision());

        imbversion = 0;
        imbversion = mvcdevice.getBootloaderVersion();
        sprintf(pimbinfo->pimbitem->bootloader_version,"%d.%d.%d.%d",
                imbversion.getVersion(),
                imbversion.getRevision(),
                imbversion.getBuildVersion(),
                imbversion.getBuildRevision());

        imbversion = 0;
        imbversion = mvcdevice.getDriverVersion();
        sprintf(pimbinfo->pimbitem->driver_version,"%d.%d.%d.%d",
                imbversion.getVersion(),
                imbversion.getRevision(),
                imbversion.getBuildVersion(),
                imbversion.getBuildRevision());

        imbversion = 0;
        imbversion = mvcdevice.getFirmwareVersion();
        sprintf(pimbinfo->pimbitem->firmware_version,"%d.%d.%d.%d",
                imbversion.getVersion(),
                imbversion.getRevision(),
                imbversion.getBuildVersion(),
                imbversion.getBuildRevision());

        uid = mvcdevice.getUID();
        sprintf(pimbinfo->pimbitem->serial_nr,"CHN-II_%06d",uid);
#endif



#if 1
        char configfilepath[BUFF_SIZE_255];
        char version_date[20];

        memset(configfilepath,0,sizeof(g_sms_configfile));
        sprintf(configfilepath,"%s",g_sms_configfile);


#if 0
        printf("GetImbInfoFromConfigFile:%s\n",m_diskconfigfilefullpath);
#endif




        ret = g_systemconfig.GetSmsVersioninfo(pimbinfo->pimbitem-
>api_version,10,

pimbinfo->pimbitem->driver_version,10,
```

```
pimbinfo->pimbitem->bootloader_version,10,

pimbinfo->pimbitem->firmware_version,10,

pimbinfo->pimbitem->version_date,20,

configfilepath);
        if(ret != SMS_SUCCESS)
        {
                return ret;
        }


        ret = g_systemconfig.GetSms_SM_Fileinfo(pimbinfo->pimbitem-
>private_file,10,

        pimbinfo->pimbitem->chain_file,10,

        configfilepath);
        if(ret != SMS_SUCCESS)
        {
                return ret;
        }

        uid = mvcdevice.getUID();
        sprintf(pimbinfo->pimbitem->serial_nr,"CHN-II_%06d",uid);

#endif


        imbs_table.UpdateItem(pimbinfo->pimbitem);

        DeleteSpaceDeviceImbInfo(&pimbinfo);

        return ret;
}
int deviceimb_proc::GetCurrentTimeFromImbBoard(time_t *pcurtime_imbboard)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;
        MvcDevice mvcdevice;
        DEVICE_IMB_INFO *pimbinfo = NULL;
        ImbsTable imbs_table;

        NewSpaceDeviceImbInfo(&pimbinfo);
        imbs_table.GetItemById(m_imbid,pimbinfo->pimbitem);

        ret = get_mvcdevice(&mvcdevice,pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        *pcurtime_imbboard = mvcdevice.getSystemPosixTime();

        DeleteSpaceDeviceImbInfo(&pimbinfo);

        return ret;

}
int deviceimb_proc::ResetImbBoard()
```

```
{
        int ret = SMS_SUCCESS;
        TMmRc mvc_ret;
        MvcDevice mvcdevice;
        DEVICE_IMB_INFO *pimbinfo = NULL;
        ImbsTable imbs_table;

        NewSpaceDeviceImbInfo(&pimbinfo);
        imbs_table.GetItemById(m_imbid,pimbinfo->pimbitem);

        ret = get_mvcdevice(&mvcdevice,pimbinfo);
        if(SMS_SUCCESS != ret)
        {
                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }

        mvc_ret = mvcdevice.resetCard();
        if(SMS_SUCCESS != ret)
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to reset
                imb board:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                DeleteSpaceDeviceImbInfo(&pimbinfo);
                return ret;
        }


        DeleteSpaceDeviceImbInfo(&pimbinfo);

        return ret;
}
int deviceimb_proc::GetDeviceState()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;

        if(!m_device_init_ok)
        {
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        mvc_ret = m_pmvcdevice->getDeviceState();
    if (MM_IS_ERROR(mvc_ret))
    {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"could not get
                device state:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        printf("could not get device state\n");
        return SMS_IMBCONTROLLER_IMB_GETSTATE_ERROR;
    }

        return ret;
```

```
}
int deviceimb_proc::Connect_Test(DEVICE_IMB_INFO *pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
    MvcDevice mvcdevice;
        CFileManager filemanger;
        int isping_ok = 0;
        ImbsTable imbs_table;

        if(NULL == pdeviceimbinfo ||
                (!(strcmp(pdeviceimbinfo->pimbitem->ip_address1,""))))
        {
                return SMS_PARAMETER_ERROR;
        }

#ifndef _WIN32
        ret = filemanger.PingIsOk(pdeviceimbinfo->pimbitem-
        >ip_address1,&isping_ok);
#else
        isping_ok = 1;
#endif

        if(isping_ok)
        {
                sprintf(pdeviceimbinfo->pimbitem->ip_index,"1");
                imbs_table.UpdateItem(pdeviceimbinfo->pimbitem);
                ret = get_mvcdevice(&mvcdevice,pdeviceimbinfo);
        }

        if((!isping_ok) ||
                SMS_SUCCESS != ret)
        {

                if(strcmp(pdeviceimbinfo->pimbitem->ip_address2,""))
                {
#ifndef _WIN32
                        ret = filemanger.PingIsOk(pdeviceimbinfo->pimbitem-
                        >ip_address2,&isping_ok);
#endif

                        if(isping_ok)
                        {
                                sprintf(pdeviceimbinfo->pimbitem-
                                >ip_index,"2");
                                imbs_table.UpdateItem(pdeviceimbinfo-
                                >pimbitem);
                                ret = get_mvcdevice(&mvcdevice,pdeviceimbinfo);
                                if(SMS_SUCCESS != ret)
                                {
                                        sprintf(pdeviceimbinfo->pimbitem-
                                        >ip_index,"2");
                                        imbs_table.UpdateItem(pdeviceimbinfo-
                                        >pimbitem);
                                }
                        }
                        else
                        {
                                sprintf(pdeviceimbinfo->pimbitem-
                                >ip_index,"1");
                                imbs_table.UpdateItem(pdeviceimbinfo-
                                >pimbitem);
                        }
```

```
                }
                else
                {
                        sprintf(pdeviceimbinfo->pimbitem->ip_index,"1");
                        imbs_table.UpdateItem(pdeviceimbinfo->pimbitem);
                }

        }


        if(!isping_ok)
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"could not ping
                imb board:%s,%s",pdeviceimbinfo->pimbitem-
                >ip_address1,pdeviceimbinfo->pimbitem->ip_address2);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return URL_IMBCONTROLLER_IMB_CONNECT_ERROR;
        }


        return ret;
}
int deviceimb_proc::ConnectedDeviceImb()
{
        int ret = SMS_SUCCESS;
    MvcDevice mvcdevice;
        TMmRc mvc_ret;


    //if (argc < 2)
    //{
    //    printf("specify filename with full path and printf-style number
    counting\n");
    //    printf("%s \"<path>\\%%08d\"\n",argv[0]);
    //    exit(0);
    //}

    // MVC20x card enumeration example
    //{
//    /// XLQ:
//#ifndef MVC2API_NETWORK_ONLY
//    MvcDeviceIterator mvcitor;
//#else
//    MvcNetDeviceIterator mvcitor(IMB_IP_ADDRESS);
//#endif
//    /// XLQ
//
//    MvcDevice mvcdev;
//
//    while(mvcdev = mvcitor.getNext())
//    {
//        printf("MVC card found: UID:%d\n",mvcdev.getUID());
//        NetworkInterfaceInfo netinfo;
//        netinfo = mvcdev.getNetworkConfiguration();
//        uint8_t ipaddr[4];
//        uint8_t mac[6];
//        netinfo.getIPAddress(ipaddr);
//        netinfo.getMACAddress(mac);
```

```
//        printf("NetworkConfiguration:%d.%d.%d.%d at MAC:
%02x-%02x-%02x-%02x-%02x-%02x\n",ipaddr[0],ipaddr[1],ipaddr[2],ipaddr[3],
//            mac[0],mac[1],mac[2],mac[3],mac[4],mac[5]);
//    }
    //}

    // use first card example



        mvc_ret = get_mvcdevice(&mvcdevice,m_pimbinfo);

#if 0
        if(!strcmp())
        {
        }

        memset(m_imb_info.connected_status,0,sizeof(m_imb_info.connected_status
        ));
        if(SMS_SUCCESS == ret)
        {
                strcpy(m_imb_info.connected_status,"true");
        }
        else
        {
                strcpy(m_imb_info.connected_status,"false");
        }
#endif


        return SMS_SUCCESS;
}
int deviceimb_proc::InitPlayDevice(DEVICE_IMB_INITPARAMETER
*pimb_initparameter)
{
        int ret = SMS_SUCCESS;

        memset(&m_imb_initparameter,0,sizeof(DEVICE_IMB_INITPARAMETER));

        memcpy(&m_imb_initparameter,pimb_initparameter,sizeof(DEVICE_IMB_INITP
        ARAMETER));

        ret = init_playdevice();
        if(SMS_SUCCESS != ret )
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"initialize imb
                board:%s %s is failed",m_pimbinfo->pimbitem-
                >ip_address1,m_pimbinfo->pimbitem->ip_address2);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);
                return ret;
        }

        return ret;
}
int deviceimb_proc::ReleasePlayDevice()
{
        int ret = SMS_SUCCESS;

        ret = uninit_playdevice();
```

```
        if(SMS_SUCCESS != ret )
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"release imb
                board:%s is failed",m_pimbinfo->pimbitem->ip_index);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);
                return ret;
        }

        return ret;
}
int deviceimb_proc::DisConnectedDeviceImb()
{
        int ret = SMS_SUCCESS;

        return ret;
}

int deviceimb_proc::ReSetDeviceImb(DEVICE_IMB_SETTING *pimbsetting)
{
        int ret = SMS_SUCCESS;

        memset(&m_imbsetting,0,sizeof(DEVICE_IMB_SETTING));
        memcpy(&m_imbsetting,pimbsetting,sizeof(DEVICE_IMB_SETTING));

#if 0
        ret = uninit_playdevice();
        if(SMS_SUCCESS !=ret )
        {
                return ret;
        }

        ret = DisConnectedDeviceImb();
        if(SMS_SUCCESS !=ret )
        {
                return ret;
        }

        ret = init_playdevice();
        if(SMS_SUCCESS !=ret )
        {
                return ret;
        }

#endif

        ret = reset_playdevice();
        if(SMS_SUCCESS != ret )
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"setting imb
                board:%s is failed",m_pimbinfo->pimbitem->ip_index);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);
                return ret;
        }

        return ret;
```

```
}
int deviceimb_proc::PlayCpl(int cplid,int enterpoint_cpl,PLAY_CONTROL
playcontrol)
{
        int ret = SMS_SUCCESS;

        sms_timeconvert timeconvert;
        char currtime_str[50];
        time_t time_start = 0;
        time_t time_end = 0;

#if 1

        printf("\n\n-----------------------------------------------------------
        ----------------\n");
        time_start = time(&time_start);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("PlayCpl enter:%s\n",currtime_str);

        printf("imbcontroller\n");
        printf("play cpl:%d\nenter point:%d\n",cplid,enterpoint_cpl);


#endif

        if(m_isplaying)
        {
                if(NULL != m_pplaybackcontrol)
                {
                        uint32_t timestamp;
                        int playstate = 0;
                        playstate = m_pplaybackcontrol->getState(&timestamp);

                        if(playstate)
                        {
                                printf("cpl auto stop and wait 1 seconds\n");
                                Stop();
                                msleep(1000);
                        }
                }
        }

        reinit_playdata();
        m_status = STATUS_PLAYING;

        m_cplid = cplid;
        m_cplframe_index = enterpoint_cpl;
        m_control = playcontrol;

        start_play();

        int runningcounter = 0;
        while(1)
        {
                //printf("stopping:m_isplaying:%d\n",m_isplaying);

                if(m_isrunning ||
                        (!m_isplaying))
                {
                        break;
                }
```

```
                runningcounter++;

                if(runningcounter > 100)
                {
                        break;
                }
                msleep(100);
        }

        printf("play timeout:%d,100\n",runningcounter);

#if 1
        time_end = time(&time_end);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("PlayCpl exit:%s\n",currtime_str);
        printf("PlayCpl seconds:%d\n",time_end-time_start);
#endif

        return ret;
}
int deviceimb_proc::StartPlayCplProcedure()
{
        int ret = SMS_SUCCESS;

        if(m_isplaying)
        {
                return ret;
        }

        m_isplaying = 1;


#if defined(_WIN32)
        HANDLE pthread_play_proc;
        pthread_play_proc = CreateThread(NULL,NULL,
        (LPTHREAD_START_ROUTINE)play_thread,this,NULL,NULL);
        if(INVALID_HANDLE_VALUE != pthread_play_proc)
        {
                ret = CloseHandle(pthread_play_proc);
        }

#else

        pthread_attr_t attr;
        unsigned long pthread_play_proc;
        pthread_attr_init(&attr);
        pthread_attr_setdetachstate (&attr, PTHREAD_CREATE_DETACHED);
        ret = pthread_create(&pthread_play_proc, &attr, play_thread, this);
//      ret = pthread_create(&pthread_play_proc, NULL, play_thread, this);
        pthread_attr_destroy (&attr);
        if( ret )
        {
                m_isplaying = 0;

                printf("play procedure thread create failed:%d\n",ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"play procedure
                thread create failed! Error code is:%d",ret);
```

```
                sprintf(m_ploginfo_insert->plog_item-
>module_name,"imbdevicecontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return ret;
        }

#endif

        m_isplaying = 1;


        return ret;
}
int deviceimb_proc::StopPlayCplProcedure()
{
        int ret = SMS_SUCCESS;

        m_isplaying = 0;

        return ret;
}
int deviceimb_proc::IsPlaying(int *pisplaying,
                                        int *pcpl_id)
{
        int ret = SMS_SUCCESS;

        if(m_isplaying)
        {
                *pisplaying = 1;
                if(NULL != m_pimbinfo)
                {
                        *pcpl_id = m_cplid;
                }
                else
                {
                        *pcpl_id = 0;
                        *pisplaying = 0;
                }
        }
        else
        {
                *pcpl_id = 0;
                *pisplaying = 0;
        }

        return ret;
}
int deviceimb_proc::GetPlayState(char *pstate,
                                                        int state_maxlen,
                                                        int *pcpl_totalframe,
                                                        int *pcpl_elapsedframe,
                                                        int *pcpl_remainframe,
                                                        int
*pcpl_elapsedseconds,
                                                        int
*pcpl_remainseconds,
                                                        int *ppercent)
{
        int ret = SMS_SUCCESS;

        if( NULL == pstate||
                NULL == pcpl_totalframe||
```

```
                NULL == pcpl_elapsedframe||
                NULL == pcpl_remainframe||
                NULL == pcpl_elapsedseconds||
                NULL == pcpl_remainseconds||
                NULL == ppercent)
        {
                return SMS_PARAMETER_ERROR;
        }

        memset(pstate,0,state_maxlen);

        if(!m_isplaying)
        {
                if( STATUS_FINISHED == m_status)
                {
                        sprintf(pstate,"finished");
                }
                else if( STATUS_FINISHED_ABORT == m_status)
                {
                        sprintf(pstate,"finished_abort");
                }
                else if( STATUS_STOPED == m_status)
                {
                        sprintf(pstate,"stoped");
                }
                else if( STATUS_ERROR == m_status)
                {
                        sprintf(pstate,"finished_error");
                }
                else
                {
                        sprintf(pstate,"unknown");

#if 1
                        printf("imbcontroller unknown");
#endif
                }

                *pcpl_totalframe = m_cplframe_count;
                *pcpl_elapsedframe = m_cplframe_index;
                *pcpl_remainframe = 0;
                *pcpl_elapsedseconds = m_cplsecond_count;
                *pcpl_remainseconds = 0;
                *ppercent = 100;
        }
        else
        {

                if( STATUS_PAUSEING == m_status)
                {
                        sprintf(pstate,"pausing");
                }
                else
                {
                        sprintf(pstate,"playing");
                }

                *pcpl_totalframe = m_cplframe_count;
                *pcpl_elapsedseconds = m_playing_seconds;
                *pcpl_remainseconds = m_cplsecond_count -
                (*pcpl_elapsedseconds);
                *pcpl_elapsedframe = (*pcpl_elapsedseconds) * m_framerate;
                *pcpl_remainframe = m_cplframe_count - (*pcpl_elapsedframe);
```

```
                if(!m_cplframe_count)
                {
                        *ppercent = 0;
                }
                else
                {
                        *ppercent = ((*pcpl_elapsedseconds) * 100) /
                m_cplsecond_count;

                }
        }


#if 0

        printf("\n++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        ++++++++++++++\n");
        printf("pstate:%s\n",pstate);
        printf("cpl total frames:%d\n",m_cplframe_count);
        printf("cpl elapsed frames:%d\n",*pcpl_elapsedframe);
        printf("cpl remain frames:%d\n",*pcpl_remainframe);
        printf("cpl total seconds:%d\n",m_cplsecond_count);
        printf("cpl elapsed seconds:%d\n",*pcpl_elapsedseconds);
        printf("cpl remain seconds:%d\n",*pcpl_remainseconds);
        printf("ppercent:%d\n",*ppercent);

        printf("\n------------------------------------------------------------
        --------------\n");
#endif


        return ret;
}
int deviceimb_proc::ResetPlayState()
{
        int ret = SMS_SUCCESS;
#if 0
        if(m_isplaying)
        {
                return URL_IMBCONTROLLER_RESETPLAYSTATE_INPLAYING_ERROR;
        }
#endif

        printf("rrrrrrrrrrrrrrrrrrrrrrrrrrrrrrrr\n");
        printf("ResetPlayState\n");

        reinit_playdata();

        return ret;

}
int deviceimb_proc::start_play()
{
        int ret = SMS_SUCCESS;

        m_isplaying = 1;

#if defined(_WIN32)
        HANDLE pthread_play_proc;
        pthread_play_proc = CreateThread(NULL,NULL,
        (LPTHREAD_START_ROUTINE)play_thread,this,NULL,NULL);
```

```
        if(INVALID_HANDLE_VALUE != pthread_play_proc)
        {
                ret = CloseHandle(pthread_play_proc);
        }

        HANDLE pthread_time_proc;
        pthread_time_proc = CreateThread(NULL,NULL,
        (LPTHREAD_START_ROUTINE)time_thread,this,NULL,NULL);
        if(INVALID_HANDLE_VALUE != pthread_time_proc)
        {
                ret = CloseHandle(pthread_time_proc);
        }

#else

        pthread_attr_t attr;
        unsigned long pthread_play_proc;
        pthread_attr_init(&attr);
        pthread_attr_setdetachstate (&attr, PTHREAD_CREATE_DETACHED);
        ret = pthread_create(&pthread_play_proc, &attr, play_thread, this);
//      ret = pthread_create(&pthread_play_proc, NULL, play_thread, this);
        pthread_attr_destroy (&attr);
        if( ret )
        {
                m_isplaying = 0;

                printf("play procedure thread create failed:%d\n",ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"play procedure
                thread create failed! Error code is:%d",ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return ret;
        }


        unsigned long pthread_time_proc;
        pthread_attr_init(&attr);
        pthread_attr_setdetachstate (&attr, PTHREAD_CREATE_DETACHED);
        ret = pthread_create(&pthread_time_proc, &attr, time_thread, this);
//      ret = pthread_create(&pthread_time_proc, NULL, time_thread, this);
        pthread_attr_destroy (&attr);
        if( ret )
        {
                m_isplaying = 0;

                printf("time procedure thread create failed:%d\n",ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"time procedure
                thread create failed! Error code is:%d",ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return ret;
        }
```

```
#endif

        return ret;
}
int deviceimb_proc::stop_play()
{
        int ret = SMS_SUCCESS;

        m_isplaying = 0;

        return ret;
}
int deviceimb_proc::play_procedure()
{
        int ret = SMS_SUCCESS;
        sms_timeconvert timeconvert;
        int diff_seconds = 0;
        TIME_FORMAT_TYPE format_type;
        time_t time_temp;
        char str_time[50];
        char video_mxf_fullpath[255];
        char audio_mxf_fullpath[255];
        char subtitle_xml_fullpath[255];
        char buff_fullpath_kdmdest[255];
        char buff_fullpath_cpldest[255];
        unsigned long mxfframe_count_video = 0;
        unsigned long mxfframe_count_audio = 0;
        int cplframe_count = 0;
        int mxfframe_count = 0;
        int cplframe_index = 0;
        int mxfframe_index = 0;
        int cache_frame_index = 0;
        int sleep_time = 0;
        ASSET_INFO *passetinfo_video = NULL;
        ASSET_INFO *passetinfo_audio = NULL;
        ASSET_INFO *passetinfo_subtitle = NULL;
        ImbsTable imbs_table;

        DEVICE_IMB_INFO deviceimb_info_master;
        MXFPARSER_VIDEO_INFO mxfparser_video_info;
        MXFPARSER_AUDIO_INFO mxfparser_audio_info;

        DEVICE_IMB_INITPARAMETER imb_initparameter;
        DEVICE_IMB_SETTING imb_setting;

        Sms_Kdm kdmcontrol;
        Sms_Cpl cpl_control;
        CPL_INFO *pcplinfo_current = NULL;
        CMXFParserModule mxfparser;

        char currtime_str[50];
        time_t time_cpl_play_start = 0;
        time_t time_cpl_play_end = 0;

        cpl_control.NewSpaceCplInfo(&pcplinfo_current);

        NewSpaceDeviceImbInfo(&m_pimbinfo);
        imbs_table.GetItemById(m_imbid,m_pimbinfo->pimbitem);

    TMmRc mvc_ret = MMRC_Ok;

        if( -1 != m_cplid )
        {
```

```
                cpl_control.GetCplInfoById(m_cplid,pcplinfo_current,1);
                format_type = TIME_FORMAT_0;
                timeconvert.GetCurrentTime(pcplinfo_current->pcpl_item-
>last_playtime,50,format_type);
                cpl_control.UpdateCplInfo(pcplinfo_current);
        }

        m_device_init_ok = 0;
        cplframe_count = 0;
        mxframe_count = 0;
        cplframe_index = 0;
        mxframe_index = 0;
        m_cached_frames_ok = 0;
        cache_frame_index = 0;

#if 1
        m_psecuritymanager = NULL;
    m_pdecoder_j2k = NULL;
    m_pdecoder_j2k_right = NULL;
        m_psubtitleDec = NULL;
        m_pdecoder_pcm = NULL;
        m_poutput_video = NULL;
        m_poutput_video_right = NULL;
        m_poutput_audio = NULL;
        m_pplaybackcontrol = NULL;

        m_isloaded = 0;
        m_isplayed = 0;
        m_ismanu_schedule_mode = 0;
        m_isshift_imb_device = 0;
        m_framedata_video_left.pbuff_data =  new unsigned
        char[MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER];
        m_framedata_video_right.pbuff_data =  new unsigned
        char[MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER];
        m_framedata_audio.pbuff_data =  new unsigned
        char[MAX_LENGTH_AUDIOFRAME_OUTPUT_BUFFER];


        m_count_cache_frame = CACHE_FRAME_COUNT;
        m_device_init_ok = 0;

#endif



#if 0
        /// IMB板卡连接检查
        ret = Connect_Test(m_pimbinfo->pimbitem->ip_address);

        if(SMS_SUCCESS != ret)
        {
                m_status = STATUS_ERROR;
                m_isplaying = 0;
        }
#endif


#if 0
        ///cpl是否可播放检查（包括完整性  kdm是否存在  及kdm是否在有效期）
        CPL_INFO *pfilter_cpl = NULL;
        CPL_INFOS cpl_infos;
        int usingstatus = 0;
        char start_at[50];
```

```
        cpl_control.NewSpaceCplInfo(&pfilter_cpl);

        strcpy(pfilter_cpl->pcpl_item->uuid,pcplinfo_current->pcpl_item->uuid);

        ret = cpl_control.GetCplInfosByFilter(&cpl_infos,pfilter_cpl,1);
        if(SMS_SUCCESS != ret ||
                0 >= cpl_infos.size() )
        {
                m_status = STATUS_ERROR;
                m_isplaying = 0;
        }

        timeconvert.GetCurrentTime(start_at,sizeof(start_at),TIME_FORMAT_2);
        cpl_control.GetCplUsingStatus(cpl_infos[0],start_at,&usingstatus);

        cpl_control.DeleteSpaceCplInfos(&cpl_infos);
        cpl_control.DeleteSpaceCplInfo(&pfilter_cpl);

        if(usingstatus)
        {
                m_status = STATUS_ERROR;
                m_isplaying = 0;
        }
#endif

        printf("play enter\n");

        while(m_isplaying)
        {
#if 1
                printf("playing\n");
#endif

                if(NULL != strstr(pcplinfo_current->pcpl_item-
                >properties,"3D"))
                {
                        g_serverstate.Set3dMode(1);
                }
                else
                {
                        g_serverstate.Set3dMode(0);
                }

                cplframe_count = atoi(pcplinfo_current->pcpl_item->duration);
                m_cplsecond_count = atoi(pcplinfo_current->pcpl_item-
                >duration_in_seconds);
                m_cplframe_count = cplframe_count;
                cplframe_index = m_cplframe_index;

                if(!m_device_init_ok)
                {
                        printf("cpl start\n");
                        //// 填充设备配置

                        memset(&imb_initparameter,0,sizeof(DEVICE_IMB_INITPARA
                        METER));
                        memset(&imb_setting,0,sizeof(DEVICE_IMB_SETTING));


                        sprintf(imb_initparameter.video_encode,"%s",pcplinfo_cu
                        rrent->pcpl_item->video_encode);
                        imb_initparameter.vedioproperty =
```

```
                        VideoOutput::VideoProperty_Dual_HDTV;

                        imb_initparameter.audio_channelcount =
                        atoi(pcplinfo_current->pcpl_item->channelcount);
                        imb_initparameter.audio_bitspersample =
                        atoi(pcplinfo_current->pcpl_item->bitspersample);
                        imb_initparameter.audio_channelmask = 0;

                        if(NULL != strstr(pcplinfo_current->pcpl_item-
                        >properties,"3D"))
                        {
                                imb_initparameter.is3D = 1;
                        }
                        else
                        {
                                imb_initparameter.is3D = 0;
                        }

                        if(NULL != strstr(pcplinfo_current->pcpl_item-
                        >properties,"Encrypted"))
                        {
                                imb_initparameter.isencrypted = 1;
                        }
                        else
                        {
                                imb_initparameter.isencrypted = 0;
                        }

                        float fframerate = 0.0;
                        char scan_mode[10];
                        memset(scan_mode,0,sizeof(scan_mode));
                        sprintf(scan_mode,"p");
                        char scan_mode_ext[10];
                        memset(scan_mode_ext,0,sizeof(scan_mode));


                        imb_setting.vedio_framerate = atoi(pcplinfo_current-
                        >pcpl_item->frame_rate_nom);
                        fframerate = imb_setting.vedio_framerate * 100;
//                              imb_setting.vedioproperty =
VideoOutput::VideoProperty_Dual_HDTV;


                        ret = GetVideoMode(atoi(pcplinfo_current->pcpl_item-
                        >stored_width),

atoi(pcplinfo_current->pcpl_item->stored_height),
                                                                fframerate,

&imb_setting.vedio_mode);

                        /////
                        int frame_rate = 0;
                        frame_rate = imb_setting.vedio_framerate;
                        sleep_time = 1000 / frame_rate / 4;
                        /////



                        if(SMS_SUCCESS != ret)
                        {
                                m_status = STATUS_ERROR;
                                m_isplaying = 0;
```

```
                break;
        }

        imb_setting.audio_sample = atoi(pcplinfo_current->pcpl_item->audio_sample_rate);
        imb_setting.audio_output_delay = atoi(m_pimbinfo->pimbitem->audio_output_delay);

        /// 可在此设置影片参数，如:
        /// 视频部分: 分辨率、帧率
        /// 音频部分: 帧率、声道数、采样率
        if(pcplinfo_current->psmskdminfos->size() > 0)
        {/// 密文相关
                SMS_KDM_INFOS smskdminfos;
                char buff_starttime[50];
                char buff_endtime[50];
                time_t time_cur;

                format_type = TIME_FORMAT_2;


                timeconvert.GetCurrentTime(buff_starttime,sizeof(buff_starttime),format_type);
                time_cur = timeconvert.GetCurrentTime();
                time_cur += atoi(pcplinfo_current->pcpl_item->duration_in_seconds);
                format_type = TIME_FORMAT_2;

                timeconvert.ConvertTimeIntToStr(time_cur,buff_endtime,sizeof(buff_endtime),format_type);


                kdmcontrol.GetValidKdmsByCpluuid(buff_starttime,buff_endtime,pcplinfo_current->pcpl_item->uuid,&smskdminfos);
                if(smskdminfos.size() <= 0)
                {
                        printf("have not valid kdm file\n");

                        kdmcontrol.DeleteSpaceKdmInfos(&smskdminfos);

                        m_status = STATUS_ERROR;
                        m_isplaying = 0;


                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"have not valid kdm file: %s",pcplinfo_current->pcpl_item->content_title_text);
                        sprintf(m_ploginfo_insert->plog_item->module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        break;
                }

                strcpy(buff_fullpath_kdmdest,smskdminfos[0]->pkdms_item_tbl->kdm_store_path);
```

```
                                strcpy(buff_fullpath_cpldest,pcplinfo_current-
>pcpl_item->filefullpath);


                                strcpy(imb_setting.cplfile_fullpath,buff_fullpa
th_cpldest);

                                strcpy(imb_setting.kdmfile_fullpath,buff_fullpa
th_kdmdest);
                                strcpy(imb_setting.cpl_uuid,pcplinfo_current-
>pcpl_item->uuid);

                                kdmcontrol.DeleteSpaceKdmInfos(&smskdminfos);
                        }

#if CHECK_DEVICE_MVC

#if 1

                        timeconvert.GetCurrentTime(currtime_str,sizeof(currtim
e_str),TIME_FORMAT_0);
                        printf("release device:%s\n",currtime_str);
#endif


                        ret = ReleasePlayDevice();
                        if(SMS_SUCCESS != ret)
                        {
                                printf("release imb board failed\n");
                                break;
                        }

#if 1
                        time_t time_init_start = 0;
                        time_t time_init_end = 0;
                        time_init_start = time(&time_init_start);

                        timeconvert.GetCurrentTime(currtime_str,sizeof(currtim
e_str),TIME_FORMAT_0);
                        printf("init device enter:%s\n",currtime_str);
#endif


                        ret = InitPlayDevice(&imb_initparameter);
                        if(SMS_SUCCESS != ret)
                        {
                                printf("initialize imb board failed\n");

#if 1
                                m_status = STATUS_ERROR;
                                m_isplaying = 0;
                                break;
#endif

#if 0


                                if(
                                (SMS_IMBCONTROLLER_IMB_INIT_VIDEOOUTPUT_ERROR)
                                == ret ||

                                        (SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODE
                                        R_ERROR) == ret)
                                {
                                        ret = ResetImbBoard();
```

```
                                        sleep(40);
                                }

                                ret = InitPlayDevice(&imb_initparameter);

                                if(SMS_SUCCESS != ret)
                                {
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;
                                        break;
                                }
#endif

                        }
#if 1
                        time_init_end = time(&time_init_end);

                        timeconvert.GetCurrentTime(currtime_str,sizeof(currtim
                        e_str),TIME_FORMAT_0);
                        printf("init device exit:%s\n",currtime_str);
                        printf("init device seconds:%d\n",time_init_end -
                        time_init_start);
#endif


                        ret = ReSetDeviceImb(&imb_setting);
                        if(SMS_SUCCESS != ret)
                        {
                                printf("setting imb board failed\n");

                        }
#endif

                        m_device_init_ok = 1;

#if 1
                        printf("setting imb play parameter\n");
#endif
                }

#if 1
                printf("cplframe_index:%d\n",cplframe_index);
                time_t time_cpl_loop_start = 0;
                time_t time_cpl_loop_end = 0;
                time_cpl_loop_start = time(&time_cpl_loop_start);

                timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),T
                IME_FORMAT_0);
                printf("cpl loop enter:%s\n",currtime_str);
#endif

                while(cplframe_index < cplframe_count)
                {
                        //printf("cpl looping\n");

                        ret =
                        GetAssetPositionByCplFramePosition(pcplinfo_current,

                        cplframe_index,

                        &passetinfo_video,

                        &passetinfo_audio,
```

```
                        &passetinfo_subtitle,

                        &mxfframe_index,

                        &mxfframe_count);

                        m_mxfframe_index = mxfframe_index;
                        m_mxfframe_count = mxfframe_count;

                        if(NULL != passetinfo_video)
                        {       /// 视频文件

                                memset(video_mxf_fullpath,0,sizeof(video_mxf_f
                                ullpath));
                                strcpy(video_mxf_fullpath,passetinfo_video-
                                >passet_item->filefullpath);

                                if(imb_initparameter.is3D)
                                {
                                        ret =
                                        mxfparser.Init3DVideoParser(video_mxf_f
                                        ullpath, mxfframe_count_video);

                                        if(SMS_SUCCESS != ret )
                                        {
                                                printf("initialize 3d video mxf
                                                parser failed\n");
                                                m_status = STATUS_ERROR;
                                                m_isplaying = 0;


                                                m_logcontrol.ZeroSpaceLog(m_plo
                                                ginfo_insert);
                                                sprintf(m_ploginfo_insert-
                                                >plog_item->level,"error");
                                                sprintf(m_ploginfo_insert-
                                                >plog_item->message,"initialize
                                                3d video mxf parser failed:
                                                %s",video_mxf_fullpath);
                                                sprintf(m_ploginfo_insert-
                                                >plog_item-
                                                >module_name,"imbdevicescontrol
                                                ler");

                                                m_logcontrol.AddLog(m_ploginfo_
                                                insert);

                                                break;
                                        }
                                }
                                else
                                {
                                        ret =
                                        mxfparser.InitVideoParser(video_mxf_ful
                                        lpath, mxfframe_count_video);

                                        if(SMS_SUCCESS != ret )
                                        {
                                                printf("initialize video mxf
                                                parser failed\n");
                                                m_status = STATUS_ERROR;
                                                m_isplaying = 0;
```

```
                                        m_logcontrol.ZeroSpaceLog(m_plo
                                        ginfo_insert);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->level,"error");
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->message,"initialize
                                        video mxf parser failed:
                                        %s",video_mxf_fullpath);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item-
                                        >module_name,"imbdevicescontrol
                                        ler");

                                        m_logcontrol.AddLog(m_ploginfo_
                                        insert);

                                        break;
                                }

                        }

#if SMS_DEBUG_PRINT
                        printf("load video ret:
                        %d=>%s=>%d\n",ret,video_mxf_fullpath,mxfframe_c
                        ount_video);
#endif

                        if(imb_initparameter.is3D)
                        {

                                memset(&mxfparser_video_info,0,sizeof(
                                mxfparser_video_info));

                                ret =
                                mxfparser.Get3DVideoInfo(mxfparser_vide
                                o_info.aspectratio,

        mxfparser_video_info.widthsize,

        mxfparser_video_info.heightsize,

        mxfparser_video_info.framerate,

        mxfparser_video_info.hmacflag,

        mxfparser_video_info.cryptoflag,

        mxfparser_video_info.buff_keyid);

                                if(SMS_SUCCESS != ret )
                                {
                                        printf("get 3d video mxf
                                        information failed\n");
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;


                                        m_logcontrol.ZeroSpaceLog(m_plo
                                        ginfo_insert);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->level,"error");
```

```
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->message,"get 3d
                                        video mxf information failed:
                                        %s",video_mxf_fullpath);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item-
                                        >module_name,"imbdevicescontrol
                                        ler");

                                        m_logcontrol.AddLog(m_ploginfo_
                                        insert);

                                        break;
                                }

                        }
                        else
                        {


                                memset(&mxfparser_video_info,0,sizeof(
                                mxfparser_video_info));

                                ret =
                                mxfparser.GetVideoInfo(mxfparser_video_
                                info.video_encode,

        mxfparser_video_info.aspectratio,

        mxfparser_video_info.widthsize,

        mxfparser_video_info.heightsize,

        mxfparser_video_info.framerate,

        mxfparser_video_info.hmacflag,

        mxfparser_video_info.cryptoflag,

        mxfparser_video_info.buff_keyid);

                                if(SMS_SUCCESS != ret )
                                {
                                        printf("get video mxf
                                        information failed\n");
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;


                                        m_logcontrol.ZeroSpaceLog(m_plo
                                        ginfo_insert);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->level,"error");
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->message,"get video
                                        mxf information failed:
                                        %s",video_mxf_fullpath);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item-
                                        >module_name,"imbdevicescontrol
                                        ler");

                                        m_logcontrol.AddLog(m_ploginfo_
```

```
                                        insert);

                                        break;
                                }
                        }

                }

                if(NULL != passetinfo_audio)
                {       /// 音频文件

                        memset(audio_mxf_fullpath,0,sizeof(audio_mxf_f
                        ullpath));
                        strcpy(audio_mxf_fullpath,passetinfo_audio-
                        >passet_item->filefullpath);
                        ret =
                        mxfparser.InitAudioParser(audio_mxf_fullpath,
                        mxfframe_count_audio);

                        if(SMS_SUCCESS != ret )
                        {
                                printf("initialize audio mxf parser
                                failed\n");
                                m_status = STATUS_ERROR;
                                m_isplaying = 0;


                                m_logcontrol.ZeroSpaceLog(m_ploginfo_in
                                sert);
                                sprintf(m_ploginfo_insert->plog_item-
                                >level,"error");
                                sprintf(m_ploginfo_insert->plog_item-
                                >message,"initialize audio mxf parser
                                failed:%s\n",passetinfo_audio-
                                >passet_item->filefullpath);
                                sprintf(m_ploginfo_insert->plog_item-
                                >module_name,"imbdevicescontroller");
                                m_logcontrol.AddLog(m_ploginfo_insert);

                                break;
                        }

#if SMS_DEBUG_PRINT
                        printf("load audio ret:
                        %d=>%s=>%d\n",ret,audio_mxf_fullpath,mxfframe_c
                        ount_audio);
#endif


                        memset(&mxfparser_audio_info,0,sizeof(mxfparse
                        r_audio_info));

                        ret =
                        mxfparser.GetAudioInfo(mxfparser_audio_info.sam
                        plingrate,

mxfparser_audio_info.channelcount,

mxfparser_audio_info.bitspersample,

mxfparser_audio_info.hmacflag,

mxfparser_audio_info.cryptoflag,
```

```
mxfparser_audio_info.buff_keyid);

                                if(SMS_SUCCESS != ret )
                                {
                                        printf("get audio mxf information
                                        failed\n");
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;


                                        m_logcontrol.ZeroSpaceLog(m_ploginfo_in
                                        sert);
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >level,"error");
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >message,"get audio mxf information
                                        failed:%s\n",passetinfo_audio-
                                        >passet_item->filefullpath);
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >module_name,"imbdevicescontroller");
                                        m_logcontrol.AddLog(m_ploginfo_insert);

                                        break;
                                }

                        }

                        if(NULL != passetinfo_subtitle)
                        { ///字幕文件

                                delete_subtitledecoder(&m_psubtitleDec);

                                int entry_point_subtitle = 0;
                                int subtitle_frame_index = 0;
                                int subtitle_frame_count = 0;

                                entry_point_subtitle =
                                atoi(passetinfo_subtitle->passet_item-
                                >entry_point);
                                subtitle_frame_count =
                                atoi(passetinfo_subtitle->passet_item-
                                >duration_in_frames);
                                subtitle_frame_index = mxfframe_index +
                                entry_point_subtitle;

                                if(subtitle_frame_index <
                                subtitle_frame_count)
                                {
                                        ret =
                                        new_subtitledecoder(m_pmvcdevice,&m_psu
                                        btitleDec);

                                   /// XLQ:将字幕数据作为缓冲提前载入
                                        unsigned char *psubtitle_data = new
                                        unsigned char[MAXLEN_SUBTITLE_FILE];
                                        unsigned int subtitle_datasize = 0;


                                        memset(psubtitle_data,0,MAXLEN_SUBTITLE
                                        _FILE);
                                        FILE *psubtitle_file =
                                        fopen(passetinfo_subtitle->passet_item-
```

```
                                >filefullpath, "rb");
        //                      subtitle_datasize =
        atoi(passetinfo_subtitle->passet_item->total_size);
                                subtitle_datasize =
                                fread(psubtitle_data, 1,
                                MAXLEN_SUBTITLE_FILE, psubtitle_file);
                                fclose(psubtitle_file);

                                uint32_t subtitle_resourceid = 0;
                                mvc_ret = m_psubtitleDec-
                                >sendSubtitleFile(psubtitle_data,
                                subtitle_datasize,
                                &subtitle_resourceid);
                                if (MM_IS_ERROR(mvc_ret))
                                {
                                        printf("failed to
                                        sendSubtitleFile:%d\n",
                                        mvc_ret);
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;

                                        delete [] psubtitle_data;
                                        psubtitle_data = NULL;


                                        m_logcontrol.ZeroSpaceLog(m_plo
                                        ginfo_insert);
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->level,"error");
                                        sprintf(m_ploginfo_insert-
                                        >plog_item->message,"failed to
                                        sendSubtitleFile:%s,size:
                                        %d",passetinfo_subtitle-
                                        >passet_item-
                                        >filefullpath,subtitle_datasize
                                        );
                                        sprintf(m_ploginfo_insert-
                                        >plog_item-
                                        >module_name,"imbdevicescontrol
                                        ler");

                                        m_logcontrol.AddLog(m_ploginfo_
                                        insert);

                                        break;
                                }
                                else
                                {
                                        printf("resourceId=%d\n",
                                        subtitle_resourceid);
                                }

                                //// XLQ:载入字体文件

                                for(ASSETS::iterator itasset =
                                passetinfo_subtitle->passet_items-
                                >begin();
                                        itasset != passetinfo_subtitle-
                                        >passet_items->end();
                                        itasset++)
                                {
                                        ASSET_ITEM *passet_item =
                                        (ASSET_ITEM *)(*itasset);
```

```
                                        memset(psubtitle_data,0,MAXLEN_
                                        SUBTITLE_FILE);
                                        psubtitle_file =
                                        fopen(passet_item-
                                        >filefullpath, "rb");
        //                              subtitle_datasize =
        atoi(passet_item->total_size);
                                        subtitle_datasize =
                                        fread(psubtitle_data, 1,
                                        MAXLEN_SUBTITLE_FILE,
                                        psubtitle_file);
                                        fclose(psubtitle_file);

                                        mvc_ret = m_psubtitleDec-
                                        >sendOverlayElement(passet_item
                                        ->filename, psubtitle_data,
                                        subtitle_datasize,
                                        subtitle_resourceid);
                                        if (MM_IS_ERROR(ret))
                                        {
                                                printf("failed to
                                                sendOverlayElement:
                                                %d\n", ret);
                                                m_status =
                                                STATUS_ERROR;
                                                m_isplaying = 0;

                                                delete []
                                                psubtitle_data;
                                                psubtitle_data = NULL;


                                                m_logcontrol.ZeroSpaceL
                                                og(m_ploginfo_insert);

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >level,"error");

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >message,"failed to
                                                sendOverlayElement:
                                                %s,size:
                                                %d",passet_item-
                                                >filename,subtitle_data
                                                size);

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >module_name,"imbdevice
                                                scontroller");

                                                m_logcontrol.AddLog(m_p
                                                loginfo_insert);

                                                break;

                                        }
                                }
```

```
                                int localtime_offset = 0;
                                /// 以毫秒为单位  后补两位作为以后扩展  负值：超前播放 正值：延迟播放


                                timeconvert.ConvertFrameToSeconds(subtitle_frame_index,atoi(passetinfo_subtitle->passet_item->frame_rate_nom),&localtime_offset);
                                localtime_offset *= 1000 * 100 * (-1);

                                ret = m_psubtitleDec->enableSubtitles(localtime_offset, SubtitleDecoder::Render_Soft_Shadows);
                                if (MM_IS_ERROR(ret))
                                {
                                        printf("failed to enableSubtitles:%d\n", ret);
                                        m_status = STATUS_ERROR;
                                        m_isplaying = 0;

                                        delete [] psubtitle_data;
                                        psubtitle_data = NULL;

                                        break;
                                }

                                if( NULL != psubtitle_data)
                                {
                                        delete [] psubtitle_data;
                                        psubtitle_data = NULL;

                                }
                        }

                }


                if(STATUS_ERROR == m_status)
                {
                        break;
                }

#if 0
                m_bshift = 0;
#endif

                while(mxfframe_index < mxfframe_count)
                {
                        //printf("mxf looping\n");

                        if(!m_cached_frames_ok)
                        {
                                /// 首先填充缓存
                                if(cache_frame_index++ == m_count_cache_frame)
                                {
                                        m_cached_frames_ok = 1;

#if 1
                                        time_cpl_play_start = time(&time_cpl_play_start);
```

```
                                        timeconvert.GetCurrentTime(curr
                                        time_str,sizeof(currtime_str),
                                        TIME_FORMAT_0);
                                        printf("cpl play enter:
                                        %s\n",currtime_str);
#endif


//                                      msleep(MAX_RUNNING_DELAY_TIME);
                                        if( NULL != m_pmvcdevice)
                                        {
                                                m_playing_seconds =
                                                cplframe_index /
                                                m_framerate;
                                        }

#if 1

                                        timeconvert.GetCurrentTime(curr
                                        time_str,sizeof(currtime_str),
                                        TIME_FORMAT_0);
                                        printf("running time:
                                        %s\n",currtime_str);
#endif

                                        m_isrunning = 1;
#if CHECK_DEVICE_MVC
                                        printf("run start\n");
                                        ret = m_pplaybackcontrol-
                                        >run();
                                        printf("run complete\n");

#if SMS_DEBUG_PRINT
                                        printf("play now\n");
#endif
                                        if (MM_IS_ERROR(ret))
                                        {
                                                printf("could not
                                                run\n");


                                                m_logcontrol.ZeroSpaceL
                                                og(m_ploginfo_insert);

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >level,"error");

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >message,"play control
                                                is failed to run");

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >module_name,"imbdevice
                                                scontroller");

                                                m_logcontrol.AddLog(m_p
                                                loginfo_insert);

                                                break;
```

```
                                        }
#endif
                                }

                        }

                        if( STATUS_PLAYING == m_status)
                        {
                                /// 用户播放
                                if( (CONTROL_PAUSE == m_control) &&
                                m_cached_frames_ok )
                                {
                                        printf("playing to pausing\n");

                                        ret= Pause();
                                }

                                if( CONTROL_STOP == m_control )
                                {
                                        printf("playing to stop\n");

                                        mxfframe_index =
                                        mxfframe_count;
                                        break;
                                }
                        }
                        else if(STATUS_PAUSEING == m_status )
                        {
                                if( CONTROL_STOP == m_control )
                                {
#if 1
#if 1
                                        printf("run start\n");
#endif

                                        ret = m_pplaybackcontrol-
                                        >run();
#if 1
                                        printf("run complete\n");
#endif
#endif


                                        mxfframe_index =
                                        mxfframe_count;

                                        printf("pausing to stop\n");

                                        break;
                                }

#if 0
                                printf("pausing\n");
#endif
                                msleep(1000);
                                continue;
                        }
                        else
                        {
                                mxfframe_index = mxfframe_count;
                                printf("reserve 2\n");
                                break;
                        }
```

```
                                if(SMS_SUCCESS != ret )
                                {
                                        break;
                                }

#if 0
                                printf("playing\n");
#endif

#if CHECK_DEVICE_MVC

                                int temp_entrypoint = 0;
                                m_framedata_video_left.datalen = 0;
                                m_framedata_video_right.datalen = 0;

                                m_framedata_audio.datalen = 0;

                                if(NULL != passetinfo_video)
                                {
                                        temp_entrypoint =
                                        atoi(passetinfo_video->passet_item-
                                        >entry_point);
                                        int mxf_frame_count_video =
                                        atoi(passetinfo_video->passet_item-
                                        >duration_in_frames);

                                        if(mxfframe_index + temp_entrypoint <
                                        mxf_frame_count_video)
                                        {
                                                if(imb_initparameter.is3D)
                                                {
                                                        ret =
                                                        mxfparser.Get3DFrameDat
                                                        a(mxfframe_index +
                                                        temp_entrypoint,

                                        (char
*)m_framedata_video_left.pbuff_data,

                                        m_framedata_video_left.datalen,

                                        m_framedata_video_left.plaintextoffset,

                                        m_framedata_video_left.sourcelength,

                                        (char
*)m_framedata_video_right.pbuff_data,

                                        m_framedata_video_right.datalen,


m_framedata_video_right.plaintextoffset,

                                        m_framedata_video_right.sourcelength);
                                                }
                                                else
                                                {
                                                        ret =
                                                        mxfparser.GetVideoFrame
                                                        Data(mxfframe_index +
                                                        temp_entrypoint,

                                        (char
```

```
*)m_framedata_video_left.pbuff_data,
                                        m_framedata_video_left.datalen,
                                        m_framedata_video_left.plaintextoffset,
                                        m_framedata_video_left.sourcelength);
                                }

                                if(SMS_SUCCESS != ret )
                                {
                                        printf("get video frame
                                        failed\n");


                                        m_logcontrol.ZeroSpaceL
                                        og(m_ploginfo_insert);

                                        sprintf(m_ploginfo_inse
                                        rt->plog_item-
                                        >level,"error");

                                        sprintf(m_ploginfo_inse
                                        rt->plog_item-
                                        >message,"mxf parser is
                                        failed to get video
                                        frame");

                                        sprintf(m_ploginfo_inse
                                        rt->plog_item-
                                        >module_name,"imbdevice
                                        scontroller");

                                        m_logcontrol.AddLog(m_p
                                        loginfo_insert);

                                        break;
                                }
                                }


                        }

                        if(NULL != passetinfo_audio)
                        {
                                temp_entrypoint =
                                atoi(passetinfo_audio->passet_item-
                                >entry_point);
                                int mxf_frame_count_audio =
                                atoi(passetinfo_audio->passet_item-
                                >duration_in_frames);

                                if(mxfframe_index + temp_entrypoint <
                                mxf_frame_count_audio)
                                {
                                        ret =
                                        mxfparser.GetAudioFrameData(mxf
                                        frame_index + temp_entrypoint,

                        (char *)m_framedata_audio.pbuff_data,

                        m_framedata_audio.datalen,
```

```
                                m_framedata_audio.plaintextoffset,

                                m_framedata_audio.sourcelength);
                                        if(SMS_SUCCESS != ret )
                                        {
                                                printf("get audio frame
                                                failed\n");


                                                m_logcontrol.ZeroSpaceL
                                                og(m_ploginfo_insert);

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >level,"error");

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >message,"mxf parser is
                                                failed to get audio
                                                frame");

                                                sprintf(m_ploginfo_inse
                                                rt->plog_item-
                                                >module_name,"imbdevice
                                                scontroller");

                                                m_logcontrol.AddLog(m_p
                                                loginfo_insert);
                                                break;
                                        }

                                }


                                ret =
                                Transfer_AVFrame(m_framedata_video_left.pbuff_d
                                ata,

                                        m_framedata_video_left.datalen,

                                        m_framedata_video_left.plaintextoffset,

                                        m_framedata_video_left.sourcelength,

                                        mxfparser_video_info.hmacflag,

                                        (unsigned char
*)mxfparser_video_info.buff_keyid,

                                        m_framedata_video_right.pbuff_data,

                                        m_framedata_video_right.datalen,

m_framedata_video_right.plaintextoffset,

                                        m_framedata_video_right.sourcelength,
```

```
                                        mxfparser_video_info.hmacflag,
                                        (unsigned char
*)mxfparser_video_info.buff_keyid,
                                        m_framedata_audio.pbuff_data,
                                        m_framedata_audio.datalen,
                                        m_framedata_audio.plaintextoffset,
                                        m_framedata_audio.sourcelength,
                                        mxfparser_audio_info.hmacflag,
                                        (unsigned char
*)mxfparser_audio_info.buff_keyid);
#endif
                                if (SMS_SUCCESS != ret)
                                {
                                        //printf("picture transfer failed
                                        (%s)\n",filename);
                                        printf("video or audio frame transfer
                                        failed\n");

                                        m_logcontrol.ZeroSpaceLog(m_ploginfo_in
                                        sert);
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >level,"error");
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >message,"imb board is failed to
                                        transfer video or audio frame");
                                        sprintf(m_ploginfo_insert->plog_item-
                                        >module_name,"imbdevicescontroller");
                                        m_logcontrol.AddLog(m_ploginfo_insert);

                                        break;
                                }
#if 0
                                if(m_bshift)
                                {
#if CHECK_DEVICE_MVC
                                        mvc_ret = m_pplaybackcontrol->stop();
#if SMS_DEBUG_PRINT
                                        printf("stop now\n");
#endif
                                        if (MM_IS_ERROR(mvc_ret))
                                        {
                                                m_logcontrol.ZeroSpaceLog(m_plo
                                                ginfo_insert);
                                                sprintf(m_ploginfo_insert-
                                                >plog_item->level,"error");
                                                sprintf(m_ploginfo_insert-
                                                >plog_item->message,"play
                                                control is failed to stop");
                                                sprintf(m_ploginfo_insert-
```

```
                                                >plog_item-
                                                >module_name,"imbdevicescontrol
                                                ler");

                                                m_logcontrol.AddLog(m_ploginfo_
                                                insert);

                                                printf("could not stop\n");
                                        }
#endif

                                        m_status = STATUS_PAUSEING;
                                        m_cached_frames_ok = 0;
                                        cache_frame_index = 0;

#if 0

#if CHECK_DEVICE_MVC
                                        ret = m_pplaybackcontrol->pause();
#endif
                                        printf("pause now %d\n",ret);
                                        if (MM_IS_ERROR(ret))
                                        {
                                                printf("could not pause\n");
                                        }

                                        m_status = STATUS_PAUSEING;
                                        printf("pause ok %d\n",ret);


#if CLEAR_CACHE

#if CHECK_DEVICE_MVC
                                        m_pdecoder_j2k->flush(0,0);

                                        m_pdecoder_pcm->flush(0,0);
#endif

                                        m_cached_frames_ok = 0;
                                        cache_frame_index = 0;
#endif

#endif

                                        break;
                                }
#endif

                                mxfframe_index++;
                                cplframe_index++;

                                m_mxfframe_index = mxfframe_index;
                                m_cplframe_index = cplframe_index;


#if 0
#ifdef _WIN32
                                Sleep(sleep_time);
#else
                                usleep(sleep_time*1000);
#endif
```

```
#endif

#if 0
//                              if(!(m_cplframe_index % 72))
//                              {
                                        char datetemp[50];

                                timeconvert.GetCurrentTime(datetemp,siz
                                eof(datetemp),TIME_FORMAT_0);

                                printf("\n\n--------------------------
                                ----------------------------------------
                                ----------\n");
                                printf("imbcontroller\n");
                                printf("mxf index:
                                %d\n",mxfframe_index);
                                printf("cpl frame index:
                                %d\n",m_cplframe_index);
                                printf("cpl id:%s\n",pcplinfo_current-
                                >pcpl_item->id);
                                printf("video file:
                                %s\n",passetinfo_video->passet_item-
                                >filefullpath);
                                printf("audio file:
                                %s\n",passetinfo_audio->passet_item-
                                >filefullpath);
                                printf("%s\n",datetemp);

//                              }
#endif


                        }/// mxfframes loop end


#if 0
                        if(mxfframe_count_video != mxfframe_count_audio)
                        {
                                int framediff = mxfframe_count_video -
                                mxfframe_count_audio;
                                if(framediff > 0)
                                {
                                        cplframe_count -= framediff;
                                }
                                else
                                {
                                        cplframe_count += framediff;
                                }
                                m_cplframe_count = cplframe_count;
                        }
#endif


                        if( m_control == CONTROL_STOP)
                        {
                                cplframe_index = cplframe_count;
                                m_cplframe_index = cplframe_index;

                                m_status = STATUS_FINISHED_ABORT;

                                break;
                        }
```

```
                        if(ret != SMS_SUCCESS)
                        {       /// 播放过程中出现错误
                                cplframe_index = cplframe_count;
                                m_cplframe_index =cplframe_index;
                                break;
                        }


                }/// cpl frames loop end

#if 1
                time_cpl_loop_end = time(&time_cpl_loop_end);

                timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),T
                IME_FORMAT_0);
                printf("cpl loop exit:%s\n",currtime_str);
                printf("cpl loop seconds:%d\n",time_cpl_loop_end -
                time_cpl_loop_start);
#endif

                if(SMS_SUCCESS != ret)
                {       /// cpl播放过程中出现错误

                        if( STATUS_FINISHED != m_status)
                        {
                                m_status = STATUS_ERROR;
                        }

                }
                else
                {       /// cpl播放成功
                        if( STATUS_PAUSEING== m_status ||
                                STATUS_PLAYING == m_status ||
                                STATUS_FINISHED_ABORT == m_status )
                        {
#if 1

#if CHECK_DEVICE_MVC
                                /// 正常结束
#if 1
                                ret = WaitForEndOfStream();
#endif

#endif
                                m_playing_seconds = atoi(pcplinfo_current-
                                >pcpl_item->duration_in_seconds);

                                if(m_status == STATUS_PLAYING)
                                {
                                        m_status = STATUS_FINISHED;
                                }

#if 1
                                time_cpl_play_end = time(&time_cpl_play_end);

                                timeconvert.GetCurrentTime(currtime_str,sizeof
                                (currtime_str),TIME_FORMAT_0);
                                printf("cpl play exit:%s\n",currtime_str);
                                printf("cpl play seconds:
                                %d\n",time_cpl_play_end - time_cpl_play_start);
#endif
```

```
#endif

#if 0
                                if(m_status == STATUS_PLAYING)
                                {
#if CHECK_DEVICE_MVC
                                        /// 正常结束
                                        ret = WaitForEndOfStream();
#endif
                                        m_status = STATUS_FINISHED;
                                }
#endif
                        }


                }/// cpl播放成功,条件判断结束

#if SMS_DEBUG_PRINT

                printf("\n\n-----------------------------------------------------
                -------------------------\n");
                printf("cpl end\n");
#endif

#if CHECK_DEVICE_MVC

#if 1
                printf("release play device enter:play procedure end\n");
#endif
                ret = ReleasePlayDevice();
#if 1
                printf("release play device exit:play procedure end\n");
#endif

#endif
                m_isplaying = 0;

        }

        DeleteSpaceDeviceImbInfo(&m_pimbinfo);

        cpl_control.DeleteSpaceCplInfo(&pcplinfo_current);

        if(NULL != m_framedata_video_left.pbuff_data)
    {
        delete[] m_framedata_video_left.pbuff_data;
        m_framedata_video_left.pbuff_data = NULL;
                m_framedata_video_left.datalen = 0;
                m_framedata_video_left.plaintextoffset = 0;
                m_framedata_video_left.sourcelength = 0;
   }

        if(NULL != m_framedata_video_right.pbuff_data)
    {
        delete[] m_framedata_video_right.pbuff_data;
        m_framedata_video_right.pbuff_data = NULL;
                m_framedata_video_right.datalen = 0;
                m_framedata_video_right.plaintextoffset = 0;
                m_framedata_video_right.sourcelength = 0;
    }
```

```
        if(NULL != m_framedata_audio.pbuff_data)
    {
        delete[] m_framedata_audio.pbuff_data;
        m_framedata_audio.pbuff_data = NULL;
                m_framedata_audio.datalen = 0;
                m_framedata_audio.plaintextoffset = 0;
                m_framedata_audio.sourcelength = 0;
    }

        m_isrunning = 0;

#if 1
        printf("play exit\n");
#endif

        return ret;
}
int deviceimb_proc::time_procedure()
{
        int ret = SMS_SUCCESS;
        sms_timeconvert timeconvert;
        time_t time_last = 0;
        time_t time_cur = 0;
        char currtime_str[50];
        uint32_t timestamp;
        int playstate = 0;
        int diff = 0;

#if 1
        printf("time enter\n");
#endif

        while(m_isplaying)
        {
#if 0
                printf("timing\n");
#endif


#if 0

                timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TI
                ME_FORMAT_0);
                printf("time procdure:%s\n",currtime_str);
#endif
                if(m_control == CONTROL_STOP)
                {
                        break;
                }

                if(NULL != m_pplaybackcontrol)
                {
                        playstate = m_pplaybackcontrol->getState(&timestamp);

#if 0
                        printf("playstate:%d,timestamp:
                        %d\n",playstate,timestamp);
#endif

                        if(1== playstate)
                        {
                                //time_cur = time(&time_cur);
                                time_cur = m_pmvcdevice->getSystemPosixTime();
```

```
                                if(!time_last)
                                {
                                        time_last = time_cur-1;
                                }

                                diff = time_cur - time_last;

                                if(diff >= 1)
                                {
                                        m_playing_seconds += diff;
                                        time_last = time_cur;
                                }
#if 0
                                printf("m_playing_seconds:
                                %d\n",m_playing_seconds);
                                printf("diff:%d\n",diff);
#endif
                                msleep(200);
                        }
                        else
                        {
                                time_cur = 0;
                                time_last = 0;
                                msleep(200);
                        }
                }
                else
                {
                        time_cur = 0;
                        time_last = 0;
                        msleep(200);
                }
        }

#if 1
        printf("time exit\n");
#endif

        return ret;
}
int deviceimb_proc::Transfer_AVFrame(const unsigned char
*pbuffer_videoframe_left,
                                        unsigned long length_videoframe_left,
                                        unsigned int
                                        uplaintextoffset_videoframe_left,
                                        unsigned int
                                        usourcelength_videoframe_left,
                                        bool bhacflag_videoframe_left,
                                        const unsigned char
                                        *ckeyid_videoframe_left,
                                        const unsigned char
                                        *pbuffer_videoframe_right,
                                        unsigned long length_videoframe_right,
                                        unsigned int
                                        uplaintextoffset_videoframe_right,
                                        unsigned int
                                        usourcelength_videoframe_right,
                                        bool bhacflag_videoframe_right,
                                        const unsigned char
                                        *ckeyid_videoframe_right,
                                        const unsigned char
                                        *pbuffer_audioframe,
                                        unsigned long length_audioframe,
```

```
                                        unsigned int
                                        uplaintextoffset_audioframe,
                                        unsigned int usourcelength_audioframe,
                                        bool bhacflag_audioframe,
                                        const unsigned char *ckeyid_audioframe)
{
        int ret = SMS_SUCCESS;

        if(!m_device_init_ok)
        {
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        if(!strcmp(m_imb_initparameter.video_encode,"mpeg2"))
        {
                ret = TransferVideo_CT(m_pmpeg2dec,

pbuffer_videoframe_left,

length_videoframe_left,

uplaintextoffset_videoframe_left,

usourcelength_videoframe_left,

bhacflag_videoframe_left,

ckeyid_videoframe_left);

                if(SMS_SUCCESS != ret )
                {
                        return ret;
                }
        }
        else
        {
                ret = TransferVideo_CT(m_pdecoder_j2k,

pbuffer_videoframe_left,

length_videoframe_left,

uplaintextoffset_videoframe_left,

usourcelength_videoframe_left,

bhacflag_videoframe_left,

ckeyid_videoframe_left);

                if(SMS_SUCCESS != ret )
                {
                        return ret;
                }

                if(m_imb_initparameter.is3D)
                {
                        ret = TransferVideo_CT(m_pdecoder_j2k_right,

pbuffer_videoframe_right,

length_videoframe_right,
```

```
uplaintextoffset_videoframe_right,

usourcelength_videoframe_right,

bhacflag_videoframe_right,

ckeyid_videoframe_right);
                        if(SMS_SUCCESS != ret )
                        {
                                return ret;
                        }
                }
        }

        ret = TransferAudio_CT(m_pdecoder_pcm,
                                                        pbuffer_audioframe,
                                                        length_audioframe,

                                                        uplaintextoffset_audiof
                                                        rame,

                                                        usourcelength_audiofram
                                                        e,
                                                        bhacflag_audioframe,
                                                        ckeyid_audioframe);
        if(SMS_SUCCESS != ret )
        {
                return ret;
        }

        return ret;
}

int deviceimb_proc::Play()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
        m_control = CONTROL_PLAY;

        if(m_isplaying)
        {
                if( STATUS_PLAYING != m_status )
                {

#if CHECK_DEVICE_MVC

                        if(NULL != m_pplaybackcontrol)
                        {
                                printf("run start\n");
                                mvc_ret = m_pplaybackcontrol->run();
                                printf("run complete\n");

                                //printf("play now\n");

                                if (MM_IS_ERROR(mvc_ret))
                                {
                                        printf("could not play\n");
                                }
                        }
#endif
                        m_status = STATUS_PLAYING;
                }
```

```
        }

        return ret;
}
int deviceimb_proc::Pause()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
        m_control = CONTROL_PAUSE;

        printf("enter Pause\n");

        if(m_isplaying)
        {

                if( STATUS_PAUSEING != m_status )
                {
                        /// 用户暂停
#if CHECK_DEVICE_MVC

                        if(NULL != m_pplaybackcontrol)
                        {
                                int playstate = 0;
                                uint32_t timeStamp;
                                playstate = m_pplaybackcontrol-
                                >getState(&timeStamp);
                                printf("playstate:%d\n",playstate);

                                if(2 != playstate)
                                {
                                        printf("pause start\n");
                                        mvc_ret = m_pplaybackcontrol->pause();
                                        printf("pause complete\n");
                                }

                                //printf("pause now\n");

                                if (MM_IS_ERROR(mvc_ret))
                                {
                                        printf("could not pause\n");
                                }
                        }
#endif
                        m_status = STATUS_PAUSEING;
                }


        }

        printf("exit Pause\n");

        return ret;
}
int deviceimb_proc::Stop()
{
        int ret = SMS_SUCCESS;
        int counter = 0;
    TMmRc mvc_ret = MMRC_Ok;
        m_control = CONTROL_STOP;

        if(m_isplaying)
        {
                if( (STATUS_PAUSEING == m_status) ||
```

```
                        (STATUS_PLAYING == m_status))
                {
                        /// 用户停止
#if CHECK_DEVICE_MVC

                        if( STATUS_PAUSEING == m_status)
                        {
                                printf("pasing to running 2\n");

                                printf("run start\n");
                                ret = m_pplaybackcontrol->run();
                                printf("run complete\n");
                        }


                        while(1)
                        {
                                //printf("stopping:m_isplaying:
                                %d\n",m_isplaying);

                                if(!m_isplaying)
                                {
                                        break;
                                }

                                counter++;

                                if(counter > 200)
                                {
                                        m_isplaying = 0;
                                        m_status = STATUS_FINISHED_ABORT;

#if 1
                                        printf("release play device
                                        enter:Stop\n");
#endif
                                        ReleasePlayDevice();
#if 1
                                        printf("release play device
                                        exit:Stop\n");
#endif

                                        break;
                                }
                                msleep(500);
                        }


                        printf("waitting timeout:%d,500\n",counter);
#endif

                }
        }

        return ret;
}
int deviceimb_proc::WaitForEndOfStream()
{
        int ret = SMS_SUCCESS;
        sms_timeconvert timeconvert;
        time_t tm_waitting_start = 0;
        time_t tm_waitting_end = 0;
        char currtime_str[50];
```

```
        if(!m_device_init_ok)
        {
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

#if 1
        tm_waitting_start = time(&tm_waitting_start);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("waitting for end enter:%s\n",currtime_str);
#endif

        if(NULL != m_pdecoder_j2k)
        {
                m_pdecoder_j2k->setEndOfStream();
        }

        if(NULL != m_pdecoder_j2k_right)
        {
                m_pdecoder_j2k_right->setEndOfStream();
        }

        if(NULL != m_pmpeg2dec)
        {
                m_pmpeg2dec->setEndOfStream();
        }

        if(NULL != m_psubtitleDec)
        {
                m_psubtitleDec->setEndOfStream();
        }

        if(NULL != m_pdecoder_pcm)
        {
                m_pdecoder_pcm->setEndOfStream();
        }

        m_pplaybackcontrol->waitForEndOfStream();


#if 1
        tm_waitting_end = time(&tm_waitting_end);

        timeconvert.GetCurrentTime(currtime_str,sizeof(currtime_str),TIME_FORM
        AT_0);
        printf("waitting for end exit:%s\n",currtime_str);
        printf("waitting seconds:%d\n",tm_waitting_end - tm_waitting_start);
#endif

        return ret;
}
int deviceimb_proc::GetVideoMode(const char *pstore_width,
                                 const char *pstore_height,
                                 int framerate,
                                 const char *pscan_mode,
                                 const char *pscan_mode_ext,
                                 VideoMode::Mode *pvideomode)
{
        int ret = SMS_SUCCESS;

        if(NULL == pstore_width ||
```

```
                NULL == pstore_height ||
                0 >= framerate ||
                NULL == pscan_mode ||
                NULL == pscan_mode_ext ||
                NULL == pvideomode ||
                (!strcmp(pstore_width,"")) ||
                (!strcmp(pstore_height,"")) ||
                (!strcmp(pscan_mode,"")))
        {
                return SMS_PARAMETER_ERROR;
        }


        if((!strcmp(pstore_width,"1920")) &&(!strcmp(pstore_height,"1080")) &&
        (6000 == framerate) &&(!strcmp(pscan_mode,"p"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_6000_p; /// 1
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (5994 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_5994_p; /// 2
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_5000_p; /// 3
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (4800 == framerate)
        &&(!strcmp(pscan_mode,"p") &&(!strcmp(pscan_mode_ext,""))))
        {
                *pvideomode = VideoMode::Mode_1920_1080_4800_p; /// 31
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (3000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_3000_p; /// 4
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2997 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2997_p; /// 5
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2500 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2500_p; /// 6
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (6000 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_6000_i; /// 7
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (5994 == framerate)
```

```
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_5994_i; /// 8
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_5000_i; /// 9
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"1250")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_5000_i_1250;    /// 10
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2400_p; /// 11
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2398_p; /// 12
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2400_psf;       /// 13
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2398_psf;       /// 14
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"1250")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2400_psf_1250;  /// 15
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"1250")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_2398_psf_1250;  /// 16
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (6000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_6000_p;  /// 17
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (5994 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_5994_p;  /// 18
```

```
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_5000_p;  /// 19
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (3000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_3000_p;  /// 20
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (2997 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_2997_p;  /// 21
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (2500 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_2500_p;  /// 22
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_2400_p;  /// 23
        }
        else if((!strcmp(pstore_width,"1280"))
        &&(!strcmp(pstore_height,"720")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1280_720_2398_p;  /// 24
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"576"))
        && (5000 == framerate) &&(!strcmp(pscan_mode,"i"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_576_5000_i;   /// 25
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"576"))
        && (2500 == framerate) &&(!strcmp(pscan_mode,"p"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_576_2500_p;   /// 26
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"576"))
        && (5000 == framerate) &&(!strcmp(pscan_mode,"p"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_576_5000_p;   /// 27
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"480"))
        && (5994 == framerate) &&(!strcmp(pscan_mode,"i"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_480_5994_i;   /// 28
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"480"))
        && (2997 == framerate) &&(!strcmp(pscan_mode,"p"))
```

```
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_480_2997_p;   /// 29
        }
        else if((!strcmp(pstore_width,"720")) &&(!strcmp(pstore_height,"480"))
        && (5994 == framerate) &&(!strcmp(pscan_mode,"p"))
        &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_720_480_5994_p;   /// 30
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (6000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_6000_p; /// 32
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (5994 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_5994_p; /// 33
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_5000_p; /// 34
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (4800 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_4800_p; /// 35
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (3000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_3000_p; /// 36
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2997 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2997_p; /// 37
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2500 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2500_p; /// 38
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2400_p; /// 39
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2398_p; /// 40
```

```
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (6000 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_6000_i; /// 41
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (5994 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_5994_i; /// 42
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (5000 == framerate)
        &&(!strcmp(pscan_mode,"i")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_5000_i; /// 43
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2400_psf;       /// 44
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"psf")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2398_psf;       /// 45
        }
        else if((!strcmp(pstore_width,"4096"))
        &&(!strcmp(pstore_height,"2160")) && (3000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_4096_2160_3000_p; /// 50
        }
        else if((!strcmp(pstore_width,"4096"))
        &&(!strcmp(pstore_height,"2160")) && (2997 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_4096_2160_2997_p; /// 51
        }
        else if((!strcmp(pstore_width,"4096"))
        &&(!strcmp(pstore_height,"2160")) && (2500 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_4096_2160_2500_p; /// 52
        }
        else if((!strcmp(pstore_width,"4096"))
        &&(!strcmp(pstore_height,"2160")) && (2400 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_4096_2160_2400_p; /// 53
        }
        else if((!strcmp(pstore_width,"4096"))
        &&(!strcmp(pstore_height,"2160")) && (2398 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_4096_2160_2398_p; /// 54
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (12000 == framerate)
```

```
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_12000_p;        /// 55
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (11988 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_11988_p;        /// 56
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (10000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_10000_p;        /// 57
        }
        else if((!strcmp(pstore_width,"1920"))
        &&(!strcmp(pstore_height,"1080")) && (9600 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_1920_1080_9600_p; /// 58
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (12000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_12000_p;        /// 59
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (11988 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_11988_p;        /// 60
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (10000 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_10000_p;        /// 61
        }
        else if((!strcmp(pstore_width,"2048"))
        &&(!strcmp(pstore_height,"1080")) && (9600 == framerate)
        &&(!strcmp(pscan_mode,"p")) &&(!strcmp(pscan_mode_ext,"")))
        {
                *pvideomode = VideoMode::Mode_2048_1080_9600_p; /// 62
        }
        else
        {
                *pvideomode = VideoMode::Mode_None;     /// 0

                float fframerate = 0.0;
                fframerate = framerate / 100.0;

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to get
                video mode:video_width:%s,video_height:%s,framerate:
                %0.2f",pstore_width,pstore_height,fframerate);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return SMS_IMBCONTROLLER_IMB_VIDEO_MODE_ERROR;
```

```
        }


        return ret;
}
int deviceimb_proc::GetVideoMode(int store_width,
                                                int store_height,
                                                int framerate,
                                                VideoMode::Mode *pvideomode)
{
        int ret = SMS_SUCCESS;

        if(0 >= store_width ||
                0 >0 == store_height ||
                0 >= framerate ||
                NULL == pvideomode)
        {
                return SMS_PARAMETER_ERROR;
        }

        if((store_width >= 2048 && store_width < 4096) ||

                (store_height >= 1080 && store_height < 2160))
        {
                *pvideomode = VideoMode::Mode_2048_1080_2400_p;
        }
        else if(store_width >= 4096 ||
                        store_height >= 2160)
        {
                *pvideomode = VideoMode::Mode_4096_2160_2400_p;
        }
        else if((store_width > 0 && store_width <= 2048) &&

                (store_height > 0 && store_height <= 1080))

        {
                *pvideomode = VideoMode::Mode_2048_1080_2400_p;
        }
        else
        {
                *pvideomode = VideoMode::Mode_None;     /// 0

                float fframerate = 0.0;
                fframerate = framerate / 100.0;

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to get
                video mode:video_width:%d,video_height:%d,framerate:
                %0.2f",store_width,store_height,fframerate);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return SMS_IMBCONTROLLER_IMB_VIDEO_MODE_ERROR;
        }


        return ret;
}
int deviceimb_proc::get_mvcdevice(MvcDevice *pmvcdevice,DEVICE_IMB_INFO
*pdeviceimbinfo)
{
```

```
        int ret = SMS_SUCCESS;
        TMmRc mvc_ret = MMRC_Ok;
        CFileManager filemanger;
        int isping_ok = 0;
        int ip_index=1;
        char ipaddress[20];

        if(NULL == pmvcdevice ||
                NULL == pdeviceimbinfo ||
                (!(strcmp(pdeviceimbinfo->pimbitem->ip_address1,"")))
        {
                return SMS_PARAMETER_ERROR;
        }

        memset(ipaddress,0,sizeof(ipaddress));

        if(!strcmp(pdeviceimbinfo->pimbitem->ip_index,"1"))
        {
                sprintf(ipaddress,"%s",pdeviceimbinfo->pimbitem->ip_address1);
        }
        else
        {
                if(strcmp(pdeviceimbinfo->pimbitem->ip_address2,""))
                {
                        sprintf(ipaddress,"%s",pdeviceimbinfo->pimbitem-
                        >ip_address2);
                }
                else
                {
                        return SMS_PARAMETER_ERROR;
                }

        }

#ifndef MVC2API_NETWORK_ONLY
    if (! ( *pmvcdevice = MvcDeviceIterator().getIndex(0) ) )
#else
        if (! ( *pmvcdevice = MvcNetDeviceIterator(ipaddress).getIndex(0) ) )
#endif
    {
        printf("failed to get imb device:%s\n",ipaddress);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to get
                imb device:%s", ipaddress);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_CONNECT_ERROR;
    }

#if 0
        mvc_ret = pmvcdevice->getDeviceState();
        if(MMRC_Ok != mvc_ret)
        {
                printf("failed to get imb board state:%s!Error code is
                %d\n",ipaddress,mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
```

```
                sprintf(m_ploginfo_insert->plog_item->message,"failed to get
                imb board state:%s!Error code is %d\n",pipaddress,mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }
#endif

        return ret;
}
int deviceimb_proc::new_securitymanager(MvcDevice *pmvcdevice,SecurityManager
**psecuritymanager,const char *pfullpath_chainfile,const char
*pfullpath_privatekeyfile)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;

        if( NULL == pmvcdevice ||
                NULL == pfullpath_chainfile ||
                (!strcmp(pfullpath_chainfile,"")) ||
                (!strcmp(pfullpath_privatekeyfile,"")) ||
                NULL == pfullpath_privatekeyfile)
        {
                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }


        delete_securitymanager(psecuritymanager);

        *psecuritymanager = new SecurityManager(&mvc_ret, *pmvcdevice);
        if(MMRC_Ok != mvc_ret)
        {
                printf("1securitymanager new is error! Error code is:%d\n",
                mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"securitymanager
                new is error! Error code is:%d", mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }
        else
        {
//              printf("1securitymanager new is success! Success code is:%d\n",
mvc_ret);
        }


        mvc_ret = (*psecuritymanager)-
        >loadCertificateChainFile(pfullpath_chainfile);
        if(MMRC_Ok != mvc_ret)
        {
                printf("2securitymanager loadcertificatechainfile is error!
                Error code is:%d\n", mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
```

```
                sprintf(m_ploginfo_insert->plog_item->message,"securitymanager
                loadcertificatechainfile is error! Error code is:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }
        else
        {
//              printf("2securitymanager loadcertificatechainfile is success!
Success code is:%d\n", mvc_ret);
        }

        mvc_ret = (*psecuritymanager)-
        >loadPrivateKeyFile(pfullpath_privatekeyfile);
        if(MMRC_Ok != mvc_ret)
        {
                printf("3SecurityManager loadprivatekeyfile is error! Error
                code is:%d\n", mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"SecurityManager
                loadprivatekeyfile is error! Error code is:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);


                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }
        else
        {
//              printf("3securitymanager loadprivatekeyfile is success! Success
code is:%d\n", mvc_ret);
        }

        mvc_ret = (*psecuritymanager)->connect();
        if(MMRC_Ok != mvc_ret)
        {
                printf("4securitymanager connect is error! Error code is:%d\n",
                mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"securitymanager
                connect is error! Error code is:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }
        else
        {
//              printf("4securitymanager connect is success! Success code is:
%d\n", mvc_ret);
        }

        mvc_ret = (*psecuritymanager)->startSuite();
        if(MMRC_Ok != mvc_ret)
        {
```

```
                printf("5securitymanager startsuite is error! Error code is:
                %d\n", mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"securitymanager
                startsuite is error! Error code is:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);


                return SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR;
        }
        else
        {
//              printf("5securitymanager startsuite is success! Success code
is:%d\n", mvc_ret);
        }


        return ret;
}
int deviceimb_proc::delete_securitymanager(SecurityManager **psecuritymanager)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;

        if(NULL == (*psecuritymanager))
        {
                return ret;
        }

        mvc_ret = (*psecuritymanager)->stopSuite();
        if(MMRC_Ok != ret)
        {
                printf("securitymanager stop suite is error! Error code is:
                %d\n", ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"securitymanager
                stop suite is error! Error code is:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);


                return -1;
        }
        else
        {
//              printf("securitymanager stop suite is success! Success code is:
%d\n", ret);
        }

        delete (*psecuritymanager);
        *psecuritymanager = NULL;

        return ret;
}
int deviceimb_proc::new_subtitledecoder(MvcDevice *pmvcdevice,SubtitleDecoder
**psubtitledec)
```

```
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;

        ret = delete_subtitledecoder(psubtitledec);

        /// 字幕解码器
        *psubtitledec = new SubtitleDecoder(&mvc_ret, *m_pmvcdevice);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to create subtitle decoder:%d\n", mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to create
                subtitle decoder:%d", mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

    // connect subtitle decoder with video output
    if (MM_IS_ERROR(mvc_ret = (*psubtitledec)-
    >connectOutput(*m_poutput_video)))
    {
        printf("failed to connect decoder with video output:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to
                connect decoder with video output:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

    // connect decoder with playback
    mvc_ret = m_pplaybackcontrol->connect(**psubtitledec);
    if (MM_IS_ERROR(ret))
    {
        printf("failed to connect playback control with subtilte decoder:
        %d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to
                connect playback control with subtilte decoder:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);


                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }


        return ret;
}
int deviceimb_proc::delete_subtitledecoder(SubtitleDecoder **psubtitledec)
```

```
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;

        if( NULL == *psubtitledec)
        {
                return ret;
        }

        if( NULL != m_poutput_video)
        {
                (*psubtitledec)->disconnectOutput(*m_poutput_video);
        }

        delete (*psubtitledec);
        *psubtitledec = NULL;


        return ret;
}
int deviceimb_proc::init_playdevice()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;

//      if(m_device_init_ok)
//      {
//              return ret;
//      }

        uninit_playdevice();

        if( NULL == m_pmvcdevice )
        {
                m_pmvcdevice = new MvcDevice();

                if(NULL == m_pmvcdevice)
                {
                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to allocate memory resource for imb board:%s is
                        failed",m_pimbinfo->pimbitem->id);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);
                        return URL_IMBCONTROLLER_ALLOC_DEVICE_RESOURCE_ERROR;
                }
        }

        mvc_ret = get_mvcdevice(m_pmvcdevice,m_pimbinfo);
        if( MMRC_Ok != mvc_ret)
        {
                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"connect imb
                board:%s is failed:%d",m_pimbinfo->pimbitem->id,mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);
                return URL_IMBCONTROLLER_IMB_CONNECT_ERROR;
        }
```

```
        /// 安全管理器
        if(m_imb_initparameter.isencrypted)
        {
                ret =
                new_securitymanager(m_pmvcdevice,&m_psecuritymanager,m_pimbinfo
                ->pimbitem->chain_file,m_pimbinfo->pimbitem->private_file);
                if(SMS_SUCCESS != ret)
                {
                        return ret;
                }
        }

    // 视频输出
        m_poutput_video = new VideoOutput(&mvc_ret, *m_pmvcdevice,
        m_imb_initparameter.vedioproperty);
        if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to create video output:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to create
                video output:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return SMS_IMBCONTROLLER_IMB_INIT_VIDEOOUTPUT_ERROR;
    }

        if(m_imb_initparameter.is3D)
        {
                m_poutput_video_right = new VideoOutput(&mvc_ret,
                *m_poutput_video);
                if (MM_IS_ERROR(ret))
                {
                        printf("failed to create right video right output:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to create right video right output:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return SMS_IMBCONTROLLER_IMB_INIT_VIDEOOUTPUT_ERROR;
                }
        }

    /// 播放控制器
    m_pplaybackcontrol = new PlaybackControl(&mvc_ret, *m_pmvcdevice);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to create playback control:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to create
                playback control:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
```

```
                m_logcontrol.AddLog(m_ploginfo_insert);


        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        if(!strcmp(m_imb_initparameter.video_encode,"mpeg2"))
        {
                m_pmpeg2dec = new Mpeg2Decoder(&mvc_ret, *m_pmvcdevice);
                if (MM_IS_ERROR(mvc_ret))
        {
                        printf("failed to create mpeg2 decoder:%d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to create mpeg2 decoder:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODER_ERROR;

        }

                if (MM_IS_ERROR(mvc_ret = m_pmpeg2dec-
                >connectOutput(*m_poutput_video)))
                {
                        printf("failed to connect decoder with video output:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to connect decoder with video output:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODER_ERROR;

                }

           // connect decoder with playback
                mvc_ret = m_pplaybackcontrol->connect(*m_pmpeg2dec);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("failed to connect playback control with mpeg2
                        decoder:%d\n",mvc_ret);


                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to connect mpeg2 control with video decoder:
                        %d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
```

```
        }
        else
        {
                /// 视频解码器
                m_pdecoder_j2k = new Jpeg2kDecoder(&mvc_ret, *m_pmvcdevice);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("failed to create Jpeg2k decoder:%d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to create Jpeg2k decoder:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODER_ERROR;
                }

                // connect j2k decoder with video output
                if(MM_IS_ERROR(mvc_ret = m_pdecoder_j2k-
                >connectOutput(*m_poutput_video)))
                {
                        printf("failed to connect decoder with video output:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to connect decoder with video output:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }

            // connect decoder with playback
                mvc_ret = m_pplaybackcontrol->connect(*m_pdecoder_j2k);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("failed to connect playback control with video
                        decoder:%d\n",mvc_ret);


                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to connect playback control with video decoder:
                        %d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }

                if(m_imb_initparameter.is3D)
                {
                        /// XLQ:视频右眼部分
                        m_pdecoder_j2k_right = new Jpeg2kDecoder(&mvc_ret,
```

```
                        *m_pdecoder_j2k);
                        if (MM_IS_ERROR(ret))
                        {
                                printf("failed to create Jpeg2k right video
                                decoder:%d\n",mvc_ret);

                                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                                sprintf(m_ploginfo_insert->plog_item-
                                >level,"error");
                                sprintf(m_ploginfo_insert->plog_item-
                                >message,"failed to create right video decoder:
                                %d",mvc_ret);
                                sprintf(m_ploginfo_insert->plog_item-
                                >module_name,"imbdevicescontroller");
                                m_logcontrol.AddLog(m_ploginfo_insert);

                                return
                                SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODER_ERROR;
                        }

                        if(MM_IS_ERROR(mvc_ret = m_pdecoder_j2k_right-
                        >connectOutput(*m_poutput_video_right)))
                        {
                                printf("failed to connect right decoder with
                                right video output:%d\n",mvc_ret);

                                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                                sprintf(m_ploginfo_insert->plog_item-
                                >level,"error");
                                sprintf(m_ploginfo_insert->plog_item-
                                >message,"failed to connect right decoder with
                                right video output:%d",mvc_ret);
                                sprintf(m_ploginfo_insert->plog_item-
                                >module_name,"imbdevicescontroller");
                                m_logcontrol.AddLog(m_ploginfo_insert);

                                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                        }

                        ret = m_pplaybackcontrol-
                        >connect(*m_pdecoder_j2k_right);
                        if (MM_IS_ERROR(ret))
                        {
                                printf("failed to connect playback control with
                                right video decoder:%d\n",mvc_ret);

                                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                                sprintf(m_ploginfo_insert->plog_item-
                                >level,"error");
                                sprintf(m_ploginfo_insert->plog_item-
                                >message,"failed to connect playback control
                                with right video decoder:%d",mvc_ret);
                                sprintf(m_ploginfo_insert->plog_item-
                                >module_name,"imbdevicescontroller");
                                m_logcontrol.AddLog(m_ploginfo_insert);


                                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                        }

                }
        }
```

```
        /// 音频解码器
        m_pdecoder_pcm = new PCMDecoder(&mvc_ret, *m_pmvcdevice,
        m_imb_initparameter.audio_bitspersample,
        m_imb_initparameter.audio_channelcount);
        if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to create pcm audio decoder:%d\n", mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to create
                pcm audio decoder:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        /// 音频输出
        m_poutput_audio = new AudioOutput(&mvc_ret, *m_pmvcdevice,
        m_imb_initparameter.audio_channelcount);
        if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to create audio output:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to create
                audio output:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

    // connect decoder with video output
    if (MM_IS_ERROR(mvc_ret = m_pdecoder_pcm-
    >connectOutput(*m_poutput_audio)))
    {
        printf("failed to connect decoder with audio output:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to
                connect decoder with audio output:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }



    mvc_ret = m_pplaybackcontrol->connect(*m_pdecoder_pcm);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("failed to connect playback control with audio decoder:
        %d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
```

```
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to
                connect playback control with audio decoder:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        m_device_init_ok = 1;

        return ret;
}
int deviceimb_proc::reset_playdevice()
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret;

        if(!m_device_init_ok)
        {
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        /// 安全管理器
        if(strcmp(m_imbsetting.kdmfile_fullpath,""))
        {
                mvc_ret = m_psecuritymanager-
                >uploadCplFile(m_imbsetting.cplfile_fullpath);
                if(MMRC_Ok != mvc_ret)
                {
                        printf("6SecurityManager is failed to upload cpl file!
                        Error code is:%d\n", mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item-
                        >message,"SecurityManager is failed to upload cpl file!
                        Error code is:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
                else
                {
                        printf("6SecurityManager is success! Success code is:
                        %d\n", mvc_ret);
                }

                mvc_ret = m_psecuritymanager-
                >uploadKdmFile(m_imbsetting.kdmfile_fullpath);
                if(MMRC_Ok != mvc_ret)
                {
                        printf("7SecurityManager is failed to upload kdm file!
                        Error code is:%d\n", mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item-
                        >message,"SecurityManager is failed to upload kdm file!
                        Error code is:%d",mvc_ret);
```

```
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);


                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
                else
                {
                        printf("7SecurityManager is success! Success code is:
                        %d\n", mvc_ret);
                }

                char authId[] = "1234567890";
//              UuidValue cplUuidArray[10] =
{"c6120e4a8e094999a195c71efc4430c8"};
                UuidValue cplUuidArray[10];

                uint32_t arrayLen = 1;
                uint64_t keyExpTime = 0;

                cplUuidArray[0] = m_imbsetting.cpl_uuid;

                mvc_ret = m_psecuritymanager->playShow(authId, cplUuidArray,
                arrayLen, &keyExpTime);
                if(MMRC_Ok != mvc_ret)
                {
                        printf("8SecurityManager is error! Error code is:%d\n",
                        mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item-
                        >message,"SecurityManager is failed to play ! Error
                        code is:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);


                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
                else
                {
                        printf("8SecurityManager is success! Success code is:
                        %d\n", mvc_ret);
                        printf("keyExpTime is:%d\n", keyExpTime);
                }

        }

        if(!strcmp(m_imb_initparameter.video_encode,"mpeg2"))
        {
        m_pmpeg2dec->setFrameRate(m_imbsetting.vedio_framerate);
                // set frame rate, so we don't need to set timestamps anymore
        }
        else
        {
                ////////////
                m_pdecoder_j2k->setFrameRate(m_imbsetting.vedio_framerate);
                                        // set frame rate, so we don't need to
                set timestamps anymore
```

```
                if(m_imb_initparameter.is3D)
                {
                        m_pdecoder_j2k_right-
                        >setFrameRate(m_imbsetting.vedio_framerate);    // set
                        frame rate, so we don't need to set timestamps anymore
                }
        }


        // setting a video mode (optional)
        if (MM_IS_ERROR(mvc_ret =  m_poutput_video-
        >setVideoMode(m_imbsetting.vedio_mode)))
    {
        printf("failed to set video mode:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to set
                video mode:%d",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);

                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        if(m_imb_initparameter.is3D)
        {
                // setting a video mode (optional)
                if (MM_IS_ERROR(mvc_ret =  m_poutput_video_right-
                >setVideoMode(m_imbsetting.vedio_mode)))
                {
                        printf("failed to set right video mode:%d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to set right video mode:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);

                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
        }

        if(strcmp(m_imbsetting.kdmfile_fullpath,""))
        {       /// 视频密文相关

                mvc_ret = m_pdecoder_j2k-
                >setSecurityManager(*m_psecuritymanager);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("j2kDecode failed to setSecurityManager:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"video
                        decoder failed to setSecurityManager:%d",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
```

```
                        m_logcontrol.AddLog(m_ploginfo_insert);


                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
        }

        if (MM_IS_ERROR(mvc_ret = m_poutput_audio-
        >setOutputFrequency(m_imbsetting.audio_sample)))
    {
        printf("failed to set audio output frequency:%d\n",mvc_ret);

                m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                sprintf(m_ploginfo_insert->plog_item->level,"error");
                sprintf(m_ploginfo_insert->plog_item->message,"failed to set
                audio output frequency:%d\n",mvc_ret);
                sprintf(m_ploginfo_insert->plog_item-
                >module_name,"imbdevicescontroller");
                m_logcontrol.AddLog(m_ploginfo_insert);


                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        if(0 != m_imbsetting.audio_output_delay)
        {
                if (MM_IS_ERROR(mvc_ret = m_poutput_audio-
                >setOutputDelay(m_imbsetting.audio_output_delay)))
                {
                        printf("failed to set audio output delay:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item->message,"failed
                        to set audio output delay:%d\n",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);


                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
        }


        if(strcmp(m_imbsetting.kdmfile_fullpath,""))
        {/// 音频密文相关
                mvc_ret = m_pdecoder_pcm-
                >setSecurityManager(*m_psecuritymanager);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("pcmDecode failed to setSecurityManager:
                        %d\n",mvc_ret);

                        m_logcontrol.ZeroSpaceLog(m_ploginfo_insert);
                        sprintf(m_ploginfo_insert->plog_item->level,"error");
                        sprintf(m_ploginfo_insert->plog_item-
                        >message,"pcmDecode failed to setSecurityManager:
                        %d\n",mvc_ret);
                        sprintf(m_ploginfo_insert->plog_item-
                        >module_name,"imbdevicescontroller");
                        m_logcontrol.AddLog(m_ploginfo_insert);
```

```
                        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
                }
        }

        return ret;
}
int deviceimb_proc::uninit_playdevice()
{
        int ret = SMS_SUCCESS;


        if(NULL != m_pplaybackcontrol)
        {

#if 0
                int playstate = 0;
                uint32_t timeStamp;
                playstate = m_pplaybackcontrol->getState(&timeStamp);

#if 1
                printf("playstate:%d,timeStamp:%d\n",playstate,timeStamp);
#endif

                if(playstate)
                {
                        if(2 == playstate)
                        {
                                m_pplaybackcontrol->run();
                        }

                        WaitForEndOfStream();
                }
#endif


                if(NULL != m_pdecoder_j2k_right)
                {
                        m_pplaybackcontrol-
                        >disconnect(*m_pdecoder_j2k_right);
                }

                if( NULL != m_pdecoder_j2k)
                {
                        m_pplaybackcontrol->disconnect(*m_pdecoder_j2k);
                }


                if( NULL != m_pmpeg2dec)
                {
                        m_pplaybackcontrol->disconnect(*m_pmpeg2dec);
                }

                if( NULL != m_psubtitleDec)
                {
                        m_pplaybackcontrol->disconnect(*m_psubtitleDec);
                }

                if( NULL != m_pdecoder_pcm)
                {
                        m_pplaybackcontrol->disconnect(*m_pdecoder_pcm);
                }
```

```
                delete m_pplaybackcontrol;
                m_pplaybackcontrol = NULL;
        }

        if(NULL != m_pdecoder_j2k_right)
        {

                if(NULL != m_poutput_video_right)
                {
                        m_pdecoder_j2k_right-
                        >disconnectOutput(*m_poutput_video_right);
                }

                delete m_pdecoder_j2k_right;
                m_pdecoder_j2k_right = NULL;
        }

        if(NULL != m_pdecoder_j2k)
        {

                if(NULL != m_poutput_video)
                {
                        m_pdecoder_j2k->disconnectOutput(*m_poutput_video);
                }

                delete m_pdecoder_j2k;
                m_pdecoder_j2k = NULL;
        }


        if(NULL != m_pmpeg2dec)
        {
                if(NULL != m_poutput_video)
                {
                        m_pmpeg2dec->disconnectOutput(*m_poutput_video);
                }

                delete m_pmpeg2dec;
                m_pmpeg2dec = NULL;
        }

        delete_subtitledecoder(&m_psubtitleDec);

        if(NULL != m_pdecoder_pcm)
        {
                if(NULL != m_poutput_audio)
                {
                        m_pdecoder_pcm->disconnectOutput(*m_poutput_audio);
                }
                delete m_pdecoder_pcm;
                m_pdecoder_pcm = NULL;
        }

        if(NULL != m_poutput_video_right)
        {
                delete m_poutput_video_right;
                m_poutput_video_right = NULL;
        }

        if(NULL != m_poutput_video)
        {
                delete m_poutput_video;
                m_poutput_video = NULL;
```

```
        }

        if(NULL != m_poutput_audio)
        {
                delete m_poutput_audio;
                m_poutput_audio = NULL;
        }

        ret = delete_securitymanager(&m_psecuritymanager);

        if(NULL != m_pmvcdevice)
        {
                delete m_pmvcdevice;
                m_pmvcdevice = NULL;
        }

        m_device_init_ok = 0;

        return ret;
}
int deviceimb_proc::transfer_audioframe(char *pbuffer_audioframe, unsigned long
length_audioframe)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
        DataBuffer databuf_audioframe;
        unsigned char *pbuff_audioframe_output = NULL;

    mvc_ret = m_pdecoder_pcm->getDataBuffer(databuf_audioframe,
    MAX_LENGTH_AUDIOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get audio databuffer audio -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }
        pbuff_audioframe_output = databuf_audioframe.getBufferAddress();
        if( NULL == pbuff_audioframe_output)
        {
        printf("could not get databuffer address audio -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }


    memcpy(pbuff_audioframe_output, pbuffer_audioframe, length_audioframe);
    mvc_ret = databuf_audioframe.send(length_audioframe);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer audio -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR;
    }

    return ret;
}
int deviceimb_proc::transfer_videoframe(char *pbuffer_videoframe, unsigned long
length_videoframe)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
    DataBuffer databuf_videoframe;
        unsigned char *pbuff_videoframe_output = NULL;
//    FILE *infile;
//
//#ifdef MM_LINUX
```

```
//    infile=fopen(filename,"rb");
//    if(infile == NULL)
//    {
//        return (-1000);
//    }
//#else
//    if (fopen_s(&infile,filename,"rb"))
//    {
//        return(-1000);
                                                    // change this to 'file not
found'
//    }
//#endif

    mvc_ret = m_pdecoder_j2k->getDataBuffer(databuf_videoframe,
    MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get databuffer video -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        pbuff_videoframe_output = databuf_videoframe.getBufferAddress();
        if( NULL == pbuff_videoframe_output)
        {
        printf("could not get databuffer address video -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

    memcpy(pbuff_videoframe_output, pbuffer_videoframe, length_videoframe);
    uint32_t readbytes = length_videoframe;

    // copy one frame here (as an example we fill the buffer with 1's)
    //uint32_t readbytes =
    static_cast<uint32_t>(fread(datbuf.getBufferAddress(),1,(size_t)
    (datbuf.getFreeSize()),infile));
    //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
    //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
    //printf("readbytes1 %d\n", readbytes);
    uint32_t padding = (16-(readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = pbuff_videoframe_output;
    for (uint32_t i=0;i<padding;i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
    mvc_ret = databuf_videoframe.send(readbytes+padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer video -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
    }


    //fclose(infile);

    //datbuf.wait(100);

        return ret;
}
```

```
int deviceimb_proc::transfer_picture(char *filename, unsigned int framecount)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
    FILE *infile = NULL;
    DataBuffer databuf_videoframe;
        unsigned char *pbuff_videoframe_output = NULL;

#ifdef MM_LINUX
    infile=fopen(filename,"rb");
    if(infile == NULL)
    {
        return (-1000);
    }
#else
    if (fopen_s(&infile,filename,"rb"))
    {
        return(-1000);
                                                // change this to 'file not
        found'
    }
#endif

    mvc_ret = m_pdecoder_j2k->getDataBuffer(databuf_videoframe,
    MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get databuffer -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        pbuff_videoframe_output = databuf_videoframe.getBufferAddress();


    // copy one frame here (as an example we fill the buffer with 1's)
    uint32_t readbytes =
    static_cast<uint32_t>(fread(pbuff_videoframe_output,1,(size_t)
    (databuf_videoframe.getFreeSize()),infile));
    //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
    //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
    //printf("readbytes1 %d\n", readbytes);
    uint32_t padding = (16-(readbytes & 0x0f)) & 0x0f;
    uint8_t *buf = pbuff_videoframe_output;
    for (uint32_t i=0;i<padding;i++)
    {
        buf[readbytes + i] = 0;
    }
    //  datbuf.setUserData(framecount);
    //printf("sending pic %d\n",framecount);
    //printf("readbytes %d\n", readbytes);
    //printf("padding %d\n", padding);
        mvc_ret = databuf_videoframe.send(readbytes+padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
    }


    fclose(infile);

    //datbuf.wait(100);
```

```
    return ret;
}
//////////
const char* deviceimb_proc::bin2hex(unsigned char* bin_buf, unsigned int
bin_len, char* str_buf, unsigned int str_len)
{
  if ( bin_buf == 0
       || str_buf == 0
       || ((bin_len * 2) + 1) > str_len )
    return 0;

//#ifdef CONFIG_RANDOM_UUID
//  const char* use_random_uuid  = getenv("KM_USE_RANDOM_UUID");
//  if ( use_random_uuid != 0 && use_random_uuid[0] != 0 && use_random_uuid[0]
!= '0' )
//    return bin2hex_rand(bin_buf, bin_len, str_buf, str_len);
//#endif

  char* p = str_buf;

  for ( unsigned int i = 0; i < bin_len; i++ )
    {
      *p = (bin_buf[i] >> 4) & 0x0f;
      *p += *p < 10 ? 0x30 : 0x61 - 10;
      p++;

      *p = bin_buf[i] & 0x0f;
      *p += *p < 10 ? 0x30 : 0x61 - 10;
      p++;
    }

  *p = '\0';
  return str_buf;
}
#if 0
int deviceimb_proc::TransferAudio_PT(char *audioDataBuffer, unsigned long
&audioDataLength)
{
        int ret = SMS_SUCCESS;
        TMmRc mvc_ret = MMRC_Ok;
        DataBuffer databuf_audioframe;
        unsigned char *pbuff_audioframe_output = NULL;

        mvc_ret = m_pdecoder_pcm->getDataBuffer(databuf_audioframe,
        MAX_LENGTH_AUDIOFRAME_OUTPUT_BUFFER);
        if (MM_IS_ERROR(mvc_ret))
        {
                printf("could not get audio databuffer -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }
        pbuff_audioframe_output = databuf_audioframe.getBufferAddress();
        if( NULL == pbuff_audioframe_output)
        {
        printf("could not get databuffer address audio -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

    memcpy(pbuff_audioframe_output, audioDataBuffer, audioDataLength);
        uint32_t readbytes = audioDataLength;

        uint32_t padding = ( 16 - (readbytes & 0x0f) ) & 0x0f;
        uint8_t *buf = pbuff_audioframe_output;
        for (uint32_t i = 0; i < padding; i++)
```

```
        {
                buf[readbytes + i] = 0;
        }
        //      datbuf.setUserData(framecount);
        //printf("sending pic %d\n",framecount);
        //printf("readbytes %d\n", readbytes);
        //printf("padding %d\n", padding);
        mvc_ret = databuf_audioframe.send(readbytes + padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer audio -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR;
    }

        //datbuf.send(audiosDataLength);
        return(ret);
}
#endif
int deviceimb_proc::TransferAudio_CT(PCMDecoder *pdecoder_pcm,
                                                                    const
unsigned char *audioDataBuffer,

unsigned long &audioDataLength,

unsigned int &uPlaintextOffset,

unsigned int &uSourceLength,
                                                                    bool
&bHmacFlag,
                                                                    const
unsigned char *cKeyID)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
        DataBuffer databuf_audioframe;
        unsigned char *pbuff_audioframe_output = NULL;

        if( 0 >= audioDataLength)
        {
                return ret;
        }

        mvc_ret = pdecoder_pcm->getDataBuffer(databuf_audioframe,
        MAX_LENGTH_AUDIOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get audio databuffer audio -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        pbuff_audioframe_output = databuf_audioframe.getBufferAddress();
        if( NULL == pbuff_audioframe_output)
        {
        printf("could not get databuffer address audio -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        uint32_t readbytes = 0;
        if( NULL == cKeyID ||
                (!strcmp((char *)cKeyID,"")))
        {       /// 明文
                memcpy(pbuff_audioframe_output, audioDataBuffer,
                audioDataLength);
```

```
                readbytes = audioDataLength;
        }
        else
        {       /// 密文

                /// XLQ:从IV段、CV段这32个字节的后面PlainText Offset段的开始取值，到
                E(V)的最后一个字节！！！
                if(true == bHmacFlag)
                {
                        memcpy( pbuff_audioframe_output,
                                (audioDataBuffer + 32),
                                (audioDataLength - 32 - 56) );

                        readbytes = audioDataLength - 32 - 56;

                        mvc_ret = databuf_audioframe.setMicValue(
                        (audioDataBuffer + (audioDataLength - 56) ), 56);
                        if (MM_IS_ERROR(mvc_ret))
                        {
                                printf("could not set mic value audio ->
                                abort\n");
                                return
                                SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR;
                        }
                }
                else
                {
                        memcpy( pbuff_audioframe_output,
                                (audioDataBuffer + 32),
                                (audioDataLength - 32) );

                        readbytes = audioDataLength - 32;
                }

                databuf_audioframe.setDecryptionSize(uPlaintextOffset,
                uSourceLength);

                char keyId[64] = "";
                bin2hex((unsigned char *)cKeyID, 16, keyId, 64);
                //printf("%s", keyId);

                mvc2::UuidValue keyid(keyId);
                //mvc2::UuidValue keyid = "25091308b27ed5bf19daec4a1b32a6be";

                databuf_audioframe.setKeyId(keyid, audioDataBuffer,
                audioDataBuffer + 16);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("could not set key id audio -> abort\n");
                        return SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR;
                }

                //databuf_audioframe.setKeyIndex(1, audioDataBuffer,
                audioDataBuffer + 16);
        }



        uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
        uint8_t *buf = pbuff_audioframe_output;
        for (uint32_t i = 0; i < padding; i++)
        {
                buf[readbytes + i] = 0;
```

```
        }

        mvc_ret = databuf_audioframe.send(readbytes + padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer audio -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR;
    }

        return(ret);
}
#if 0
int deviceimb_proc::TransferVideo_PT(char *videoDataBuffer, unsigned long
&videoDataLength)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
    DataBuffer databuf_videoframe;
        unsigned char *pbuff_videoframe_output = NULL;


        mvc_ret = m_pdecoder_j2k->getDataBuffer(databuf_videoframe,
        MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get databuffer video -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }
        pbuff_videoframe_output = databuf_videoframe.getBufferAddress();
        if( NULL == pbuff_videoframe_output)
        {
        printf("could not get databuffer address video -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        memcpy(pbuff_videoframe_output, videoDataBuffer, videoDataLength);
        uint32_t readbytes = videoDataLength;

        // copy one frame here (as an example we fill the buffer with 1's)
        //uint32_t readbytes =
        static_cast<uint32_t>(fread(datbuf.getBufferAddress(),1,(size_t)
        (datbuf.getFreeSize()),infile));
        //int readbytes = fread(datbuf.getBufferAddress(), 1, 1301870, infile);
        //printf("datbuf.getFreeSize() %d\n", datbuf.getFreeSize());
        //printf("readbytes1 %d\n", readbytes);
        uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
        uint8_t *buf = pbuff_videoframe_output;
        for (uint32_t i = 0; i < padding; i++)
        {
                buf[readbytes + i] = 0;
        }
        //      datbuf.setUserData(framecount);
        //printf("sending pic %d\n",framecount);
        //printf("readbytes %d\n", readbytes);
        //printf("padding %d\n", padding);
        mvc_ret = databuf_videoframe.send(readbytes + padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer video -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
    }

        //fclose(infile);
```

```
        //datbuf.wait(100);

        return(ret);
}
#endif
int deviceimb_proc::TransferVideo_CT(MvcDecoder *pdecoder,
                                                                    const
unsigned char *videoDataBuffer,

unsigned long &videoDataLength,

unsigned int &uPlaintextOffset,

unsigned int &uSourceLength,
                                                                    bool
&bHmacFlag,
                                                                    const
unsigned char *cKeyID)
{
        int ret = SMS_SUCCESS;
    TMmRc mvc_ret = MMRC_Ok;
    DataBuffer databuf_videoframe;
        unsigned char *pbuff_videoframe_output = NULL;

        if( 0 >= videoDataLength )
        {
                return ret;
        }

        mvc_ret = pdecoder->getDataBuffer(databuf_videoframe,
        MAX_LENGTH_VIDEOFRAME_OUTPUT_BUFFER);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not get databuffer video -> abort\n");
        return URL_IMBCONTROLLER_IMB_INIT_ERROR;
    }

        pbuff_videoframe_output = databuf_videoframe.getBufferAddress();
        if( NULL == pbuff_videoframe_output)
        {
        printf("could not get databuffer address video -> abort\n");
                return URL_IMBCONTROLLER_IMB_INIT_ERROR;
        }

        uint32_t readbytes = 0;

        if( NULL == cKeyID ||
                (!strcmp((char *)cKeyID,"")))
        {       /// 明文
                memcpy(pbuff_videoframe_output, videoDataBuffer,
                videoDataLength);
                readbytes = videoDataLength;
        }
        else
        {       /// 密文

                /// XLQ:从IV段、CV段这32个字节的后面PlainText Offset段的开始取值，到
                E(V)的最后一个字节！！！
                if(true == bHmacFlag)
                {
                        memcpy( pbuff_videoframe_output,
                                (videoDataBuffer + 32),
```

```
                                (videoDataLength - 32 - 56) );

                        readbytes = videoDataLength - 32 - 56;

                        mvc_ret = databuf_videoframe.setMicValue(
                        (videoDataBuffer + (videoDataLength - 56) ), 56);
                        if (MM_IS_ERROR(mvc_ret))
                        {
                                printf("could not set mic value video ->
                                abort\n");
                                return
                                SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
                        }

                }
                else
                {
                        memcpy( pbuff_videoframe_output,
                                (videoDataBuffer + 32),
                                (videoDataLength - 32) );

                        readbytes = videoDataLength - 32;
                }

                databuf_videoframe.setDecryptionSize(uPlaintextOffset,
                uSourceLength);

                char keyId[64] = "";
                bin2hex((unsigned char *)cKeyID, 16, keyId, 64);
                //printf("%s", keyId);

                mvc2::UuidValue keyid(keyId);
                //mvc2::UuidValue keyid = "007203b983345b84bcb0c928e8bab01b";

                mvc_ret = databuf_videoframe.setKeyId(keyid, videoDataBuffer,
                videoDataBuffer + 16);
                if (MM_IS_ERROR(mvc_ret))
                {
                        printf("could not set key id video -> abort\n");
                        return SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
                }

                //dataBuffer.setKeyIndex(0, videoDataBuffer, videoDataBuffer +
                16);
        }

        uint32_t padding = (16 - (readbytes & 0x0f)) & 0x0f;
        uint8_t *buf = pbuff_videoframe_output;
        for (uint32_t i = 0; i < padding; i++)
        {
                buf[readbytes + i] = 0;
        }
        //      databuf.setUserData(framecount);
        //printf("sending pic %d\n",framecount);
        //printf("readbytes %d\n", readbytes);
        //printf("padding %d\n", padding);
        mvc_ret = databuf_videoframe.send(readbytes + padding);
    if (MM_IS_ERROR(mvc_ret))
    {
        printf("could not send databuffer video -> abort\n");
        return SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR;
    }
        //fclose(infile);
```

```
        //datbuf.wait(100);

        return(ret);
}
int deviceimb_proc::TestPlay()
{
        int ret = SMS_SUCCESS;

#if 0
        int error_code = 0;
        char video_mxf_fullpath[255];
        char audio_mxf_fullpath[255];
        unsigned long mxfframe_count_video = 0;
        unsigned long mxfframe_count_audio = 0;
        int mxfframe_count = 0;
        int mxfframe_index = 0;
        int cache_frame_index = 0;

        memset(video_mxf_fullpath,0,sizeof(video_mxf_fullpath));
        memset(audio_mxf_fullpath,0,sizeof(audio_mxf_fullpath));

#if defined(_WIN32)
        strcpy(video_mxf_fullpath,"e:
        \\dcps\\dieying3_xyz_sub\\dieying3_xyz_video.mxf");

        strcpy(audio_mxf_fullpath,"e:
        \\dcps\\dieying3_xyz_sub\\dieying3_xyz_audio.mxf");
#else
        strcpy(video_mxf_fullpath,"/srv/diying3-xyz/dieying3_xyz_video.mxf");

        strcpy(audio_mxf_fullpath,"/srv/diying3-xyz/dieying3_xyz_audio.mxf");
#endif

        m_schedule_running = 1;
        m_cached_frames_ok = 0;
        m_device_init_ok = 0;
        m_playing_device_ok = 0;
        m_count_cache_frame = CACHE_FRAME_COUNT;

        while(m_schedule_running)
        {

                if(!m_device_init_ok)
                {

#if CHECK_DEVICE_MVC

                        ret = ReInitDevice();
                        if( SMS_SUCCESS != ret )
                        {
                                printf("screening proc:waitting device
                                initialize\n");
                                sleep(1);
                                continue;
                        }

#endif


                        m_device_init_ok = 1;
                }
```

```
                if(strcmp(video_mxf_fullpath,""))
                {
                        ret = mxfparser.InitVideoParser(video_mxf_fullpath,
                        mxfframe_count_video);


#if SMS_DEBUG_PRINT
                        printf("load video ret:
                        %d=>%s=>%d\n",ret,video_mxf_fullpath,mxfframe_count_vid
                        eo);
#endif
                }


                if(strcmp(audio_mxf_fullpath,""))
                {
                        ret = mxfparser.InitAudioParser(audio_mxf_fullpath,
                        mxfframe_count_audio);


#if SMS_DEBUG_PRINT
                        printf("load audio ret:
                        %d=>%s=>%d\n",ret,audio_mxf_fullpath,mxfframe_count_aud
                        io);
#endif

                }

                mxfframe_count = mxfframe_count_video;

                while(mxfframe_index < mxfframe_count)
                {

                        if(!m_playing_device_ok)
                        {
#if CHECK_DEVICE_MVC
                                ret = init_playdevice();
#endif
                                if(SMS_SUCCESS != ret)
                                {
                                        sleep(1);
                                        continue;
                                }
                                m_playing_device_ok = 1;
                        }


                        if(!m_cached_frames_ok)
                        {
                                /// 首先填充缓存
                                if(cache_frame_index++ == m_count_cache_frame)
                                {
                                        m_cached_frames_ok = 1;

#if CHECK_DEVICE_MVC
                                        ret = m_pplaybackcontrol->run();
#endif
                                        printf("play now %d\n",ret);
                                        if (MM_IS_ERROR(ret))
                                        {
```

```
                                        printf("could not play\n");
                                }

                        }

                }


#if CHECK_DEVICE_MVC

                mxfparser.GetVideoFrameData(mxfframe_index,
                m_pbuff_videoframe, m_length_videoframe);

                //ret = transferPic(*m_pdecoder_j2k, filename,
                framecounter);
                ret = transfer_videoframe(m_pdecoder_j2k,
                m_pbuff_videoframe, m_length_videoframe);
#endif
                if (SMS_SUCCESS != ret)
                {
                        error_code = 2;
                        break;
                }

                /// XLQ:
                //fread(audioDataBuffer, 1, 36000, fp);

#if CHECK_DEVICE_MVC
                mxfparser.GetAudioFrameData(mxfframe_index,
                m_pbuff_audioframe, m_length_audioframe);
                ret = transfer_audioframe(m_pdecoder_pcm,
                m_pbuff_audioframe,m_length_audioframe);

#endif

                if (SMS_SUCCESS != ret)
                {
                        //printf("picture transfer failed
                        (%s)\n",filename);
                        error_code = 3;
                        printf("picture transfer failed\n");
                        break;
                }


                mxfframe_index++;


#if !CHECK_DEVICE_MVC

#if defined(_WIN32)
                Sleep(40);
#else
                sleep(1);
#endif

#endif


        }/// mxfframes loop end


#if CHECK_DEVICE_MVC
```

```
                /// 正常结束

                if(m_playing_device_ok)
                {
                        m_pdecoder_j2k->setEndOfStream();

                        m_pdecoder_pcm->setEndOfStream();

                        m_pplaybackcontrol->waitForEndOfStream();

                        printf("End of stream reached\n");

                        ret = uninit_playdevice();
                }


#endif


                mxfframe_count = 0;
                mxfframe_index = 0;
                m_cached_frames_ok = 0;
                cache_frame_index = 0;

                m_schedule_running = 0;

                m_device_init_ok = 0;
                m_playing_device_ok = 0;


        }

#endif

        return ret;
}
int deviceimb_proc::reinit_playdata()
{
        int ret = SMS_SUCCESS;

        m_cached_frames_ok = 0;
        m_control = CONTROL_STOP;
        m_status = STATUS_STOPED;

        m_isloaded = 0;
        m_isplayed = 0;

        m_framerate = 24;
        m_reelinfo_index = 0;
        m_cplframe_count = 0;
        m_cplsecond_count = 0;
        m_mxfframe_count = 0;
        m_mxfframe_index = 0;

        m_playing_seconds = 0;
        m_isrunning = 0;

        return ret;
}
int deviceimb_proc::GetAssetPositionByCplFramePosition(CPL_INFO *pcplinfo,

                                int cplframe_position,
```

```
                                ASSET_INFO **passetinfo_video,

                                ASSET_INFO **passetinfo_audio,

                                ASSET_INFO **passetinfo_subtitle,

                                int *pmxfframe_index,

                                int *pmxfframecount)
{
        int ret = SMS_SUCCESS;
        int cplframe_index = 0;
        int bfind = 0;
        int mxfframe_index = 0;
        int mxfframe_count = 0;

        if( NULL == pcplinfo ||
                cplframe_position < 0 ||
                NULL == pmxfframe_index ||
                NULL == pmxfframecount)
        {
                return SMS_PARAMETER_ERROR;
        }

        if( cplframe_position >= atoi(pcplinfo->pcpl_item->duration))
        {
                return SMS_PARAMETER_ERROR;
        }

        for(REEL_INFOS::iterator itreel = pcplinfo->preelinfos->begin();
                itreel != pcplinfo->preelinfos->end(); itreel++)
        {
                        REEL_INFO *preelinfo = NULL;
                        ASSET_INFO *passetinfo_video_temp = NULL;
                        ASSET_INFO *passetinfo_audio_temp = NULL;
                        ASSET_INFO *passetinfo_subtitle_temp = NULL;
                        int max_mxf_frame_count = 0;
                        int temp_frame = 0;
                        int temp_entrypoint = 0;

                        preelinfo = (REEL_INFO *)(*itreel);
                        mxfframe_count = 0;

                        for(ASSET_INFOS::iterator itasset = preelinfo-
                        >passetinfos->begin();
                                itasset != preelinfo->passetinfos->end();
                                itasset++)
                        {
                                ASSET_INFO *passetinfo = (ASSET_INFO *)
                                (*itasset);
                                if(!strcmp(passetinfo->passet_item-
                                >asset_type,"MainPicture"))
                                {
                                        passetinfo_video_temp = passetinfo;

                                        temp_frame =
                                        atoi(passetinfo_video_temp-
                                        >passet_item->duration);
                                        temp_entrypoint =
                                        atoi(passetinfo_video_temp-
                                        >passet_item->entry_point);
                                        temp_frame -= temp_entrypoint;
```

```
                        if( temp_frame > mxfframe_count)
                        {
                                mxfframe_count = temp_frame;
                        }

                }
                else if(!strcmp(passetinfo->passet_item-
                >asset_type,"MainSound"))
                {
                        passetinfo_audio_temp = passetinfo;

                        temp_frame =
                        atoi(passetinfo_audio_temp-
                        >passet_item->duration);
                        temp_entrypoint =
                        atoi(passetinfo_audio_temp-
                        >passet_item->entry_point);
                        temp_frame -= temp_entrypoint;

                        if( temp_frame > mxfframe_count)
                        {
                                mxfframe_count = temp_frame;
                        }

                }
                else if(!strcmp(passetinfo->passet_item-
                >asset_type,"MainSubtitle"))
                {
                        passetinfo_subtitle_temp = passetinfo;

                        temp_frame =
                        atoi(passetinfo_subtitle_temp-
                        >passet_item->duration);
                        temp_entrypoint =
                        atoi(passetinfo_subtitle_temp-
                        >passet_item->entry_point);
                        temp_frame -= temp_entrypoint;

                        if( temp_frame > mxfframe_count)
                        {
                                mxfframe_count = temp_frame;
                        }

                }
                else
                {
                        continue;
                }
        }

        if(cplframe_position < cplframe_index + mxfframe_count
        )
        {
                bfind = 1;
                mxfframe_index = cplframe_position -
                cplframe_index;
                cplframe_index += mxfframe_index;
                *passetinfo_video = passetinfo_video_temp;
                *passetinfo_audio = passetinfo_audio_temp;
                *passetinfo_subtitle =
                passetinfo_subtitle_temp;
                break;
        }
```

```
                        else
                        {
                                cplframe_index += mxfframe_count;
                        }

        }  //// loop reel end



        if(!bfind)
        {
                return SMS_PARAMETER_ERROR;
        }

        *pmxfframe_index = mxfframe_index;
        *pmxfframecount = mxfframe_count;

#if SMS_DEBUG_PRINT
        printf("mxfframe_index=>%d\n",mxfframe_index);
        printf("mxfframe_count=>%d\n",mxfframe_count);
#endif


        return ret;
}
//////////
int deviceimb_proc::NewSpaceDeviceImbInfo(DEVICE_IMB_INFO **pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;
//      Sms_Cpl cpl_control;

        if(NULL != *pdeviceimbinfo)
        {
                return ret;
        }

        *pdeviceimbinfo = new DEVICE_IMB_INFO;

        (*pdeviceimbinfo)->pimbitem = NULL;

//      (*pdeviceimbinfo)->pcplinfo = NULL;
//      cpl_control.NewSpaceCplInfo(&((*pdeviceimbinfo)->pcplinfo));

        imbs_table.NewSpaceItem(&((*pdeviceimbinfo)->pimbitem));


        return ret;
}
int deviceimb_proc::DeleteSpaceDeviceImbInfo(DEVICE_IMB_INFO **pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;
//      Sms_Cpl cpl_control;

        if( NULL == *pdeviceimbinfo)
        {
                return ret;
        }


        imbs_table.DeleteSpaceItem(&((*pdeviceimbinfo)->pimbitem));
```

```
//      if(NULL != (*pdeviceimbinfo)->pcplinfo)
//      {
//              cpl_control.DeleteSpaceCplInfo(&((*pdeviceimbinfo)->pcplinfo));
//      }


        if(NULL != (*pdeviceimbinfo))
        {
                delete (*pdeviceimbinfo);
                *pdeviceimbinfo = NULL;
        }


        return ret;
}
int deviceimb_proc::ZeroSpaceDeviceImbInfo(DEVICE_IMB_INFO *pdeviceimbinfo)
{
        int ret = SMS_SUCCESS;
        ImbsTable imbs_table;
//      Sms_Cpl cpl_control;

        if( NULL == pdeviceimbinfo)
        {
                return ret;
        }

        imbs_table.ClearSpaceItem(pdeviceimbinfo->pimbitem);

//      cpl_control.ZeroSpaceCplInfo(pdeviceimbinfo->pcplinfo);

        return ret;
}
int deviceimb_proc::DeleteSpaceDeviceImbInfos(DEVICE_IMB_INFOS
*pdeviceimbinfos)
{
        int ret = SMS_SUCCESS;

        if(NULL == pdeviceimbinfos)
        {
                return ret;
        }

        if(pdeviceimbinfos->size() > 0)
        {
                for(int i = 0; i<pdeviceimbinfos->size(); i++)
                {
                        DEVICE_IMB_INFO *ptemp = (DEVICE_IMB_INFO *)
                        ((*pdeviceimbinfos)[i]);
                        DeleteSpaceDeviceImbInfo(&ptemp);
                }

                pdeviceimbinfos->clear();
        }


        return ret;
}
int deviceimb_proc::ZeroSpaceDeviceImbInfos(DEVICE_IMB_INFOS *pdeviceimbinfos)
{
        int ret = SMS_SUCCESS;

        if(NULL == pdeviceimbinfos)
        {
```

```
                return ret;
        }

        if(pdeviceimbinfos->size() > 0)
        {
                for(int i = 0; i<pdeviceimbinfos->size(); i++)
                {
                        DEVICE_IMB_INFO *ptemp = (DEVICE_IMB_INFO *)
                        ((*pdeviceimbinfos)[i]);
                        ZeroSpaceDeviceImbInfo(ptemp);
                }
        }


        return ret;
}
int deviceimb_proc::GetErrorString(int errorcode,REPORT_STATUS *preportstatus)
{
        int ret = SMS_SUCCESS;

        memset(preportstatus,0,sizeof(REPORT_STATUS));

        switch(errorcode)
        {
        case SMS_PARAMETER_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"input parameter is error");
                break;
        case DB_CONNECTED_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to connect database");
                break;
        case DB_DISCONNECTED_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to disconnect
                database");
                break;
        case URI_COMMAND_PARSE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to parse uri command");
                break;
        case URI_COMMAND_REQUEST_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to send uri request
                command");
                break;
        case URL_ADDRESS_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"url is not found");
                break;
        case URL_ERROR_CODE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get this error
                code");
                break;
        case URL_IMBCONTROLLER_NO_IMBID_ERROR:
```

```
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"have not this imb id");
                break;
        case URL_IMBCONTROLLER_IMB_CONNECT_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to connect imb");
                break;
        case URL_IMBCONTROLLER_IMB_INIT_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to init imb");
                break;
        case URL_IMBCONTROLLER_IMB_SET_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to set imb");
                break;
        case URL_IMBCONTROLLER_IMB_GETSECURITYLOG_INPLAYING_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb security log
        in playing");
                break;
        case URL_IMBCONTROLLER_GETCERT_INPLAYING_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb cert file in
        playing");
                break;
        case SMS_IMBCONTROLLER_IMB_GETSTATE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb state");
                break;
        case SMS_IMBCONTROLLER_IMB_TRANS_VIDEO_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to trans video frame");
                break;
        case SMS_IMBCONTROLLER_IMB_TRANS_AUDIO_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to trans audio frame");
                break;
        case SMS_IMBCONTROLLER_IMB_VIDEO_MODE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"video mode is invalid");
                break;
        case SMS_IMBCONTROLLER_IMB_INIT_SM_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to initilize security
        manager");
                break;
        case SMS_IMBCONTROLLER_IMB_INIT_VIDEODECODER_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to initilize video
        decoder");
                break;
```

```
        case URL_IMBCONTROLLER_GET_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb");
                break;
        case URL_IMBCONTROLLER_POST_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to post imb");
                break;
        case URL_IMBCONTROLLER_PUT_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to put imb");
                break;
        case URL_IMBCONTROLLER_DELETE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to delete imb");
                break;
        case URL_IMBCONTROLLER_IMB_RUN_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to run imbboard");
                break;
        case URL_IMBCONTROLLER_IMB_PAUSE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to run imbboard");
                break;
        case URL_IMBCONTROLLER_IMB_STOP_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to stop imbboard");
                break;
        case URL_IMBCONTROLLER_IMB_GETSECURITYLOG_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb security
                log");
                break;
        case URL_IMBCONTROLLER_GETCERT_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb cert file");
                break;
        case URL_IMBCONTROLLER_DEVICE_USING_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to operate,imb board is
                be using");
                break;
        case URL_IMBCONTROLLER_RESETPLAYSTATE_INPLAYING_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to reset playstate in
                playing");
                break;
        case URL_IMBCONTROLLER_ALLOC_DEVICE_RESOURCE_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to allocate memory
                resource for device");
```

```
                break;
        case URL_IMBCONTROLLER_IMB_GETSECURITYLOG_EMPTY_ERROR:
                sprintf(preportstatus->status,"failed");
                sprintf(preportstatus->error_code_str,"%d",errorcode);
                sprintf(preportstatus->message,"failed to get imb security
                log,log is empty");
                break;
        default:
                sprintf(preportstatus->status,"successfull");
                sprintf(preportstatus->error_code_str,"0");
                sprintf(preportstatus->message,"is successfull");
                break;
        }


        return ret;
}
```