经典服务器获取设备列表

目录

[1：结构体定义 1](#_Toc519763402)

[2：请求流程 2](#_Toc519763403)

[2.1：请求session： 2](#_Toc519763404)

[2.2登陆： 2](#_Toc519763405)

[2.3请求设备列表： 2](#_Toc519763406)

[3:代码参考 3](#_Toc519763407)

# 1：结构体定义

struct signaling\_channel\_head\_t

{

uint32\_t cmd;

uint32\_t result;

uint32\_t session;

uint32\_t datalen;

};

struct ST\_CLASSIC\_DEVICE\_INFO\_BASE

{

int dev\_is\_publish; //0:私有设备 1：publish device

int dev\_online; //是否在线

int dev\_browse\_num; //浏览用户数量

//

int dev\_id; //设备在数据库的id

int dev\_areaid; //区域id

int dev\_parent\_areaid; //区域id的父id

//

char dev\_name[64]; //设备名称

char dev\_area\_name[64]; //区域名称

char dev\_media\_ip[16]; //核心服务器ip

char reserve[88];

};

# 2：请求流程

## 2.1：请求session：

发送：signaling\_channel\_head\_t

cmd： SINGNALING\_CHANNEL\_CMD\_LOGIN\_NEED\_SESSION

服务器返回：

返回：signaling\_channel\_head\_t

cmd：SINGNALING\_CHANNEL\_CMD\_LOGIN\_NEED\_SESSION

session：服务器填充了session

## 2.2登陆：

发送：signaling\_channel\_head\_t + signaling\_channel\_user\_login\_t

signaling\_channel\_head\_t .cmd：SINGNALING\_CHANNEL\_CMD\_LOGIN

signaling\_channel\_user\_login\_t. userdes:使用des混合加密用户名与服务器返回的session

signaling\_channel\_user\_login\_t. pwddes:使用des混合加密密码与服务器返回的session

服务器返回：

返回：signaling\_channel\_head\_t

cmd：SINGNALING\_CHANNEL\_CMD\_LOGIN

result：SINGNALING\_CHANNEL\_RESULT\_OK

## 2.3请求设备列表：

发送：signaling\_channel\_head\_t

cmd： SINGNALING\_CHANNEL\_CMD\_REDIRECTION\_CLASSIC\_GET\_DEVICES

服务器返回：

返回：signaling\_channel\_head\_t + （ST\_CLASSIC\_DEVICE\_INFO\_BASE\*N）

signaling\_channel\_head\_t .cmd：

SINGNALING\_CHANNEL\_CMD\_REDIRECTION\_CLASSIC\_GET\_DEVICES

Datalen：（ST\_CLASSIC\_DEVICE\_INFO\_BASE\*N） N>=0

# 3:代码参考

XInternetProjectClient\\_Include 的 GetDevices

**void** CXClassicClientInstance::GetDevices(**char** \* pdstBuffer, **int** &len)

{

    m\_cs.Lock();

    len = 0;

    CXNetStream\* m\_pTcpStream = NULL;

    ST\_SIGNALING\_SEND\_BUFFER \*pSendBuffer = CSignalingBufferPool::Instance().malloc();;

    signaling\_channel\_head\_t msg\_key\_recv;

    do

    {

*//--------------------------------------------------------------------------------------------------------------------------*

        m\_pTcpStream = XNetCreateStream4Connect(m\_szCenterIp, 6499, 10);

        if (0 != XNetConnectStream(m\_pTcpStream))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- conn svr error");*

            break;

        }

*//--------------------------------------------------------------------------------------------------------------------------*

*//NEED\_SESSIONID*

        memset(pSendBuffer, 0, sizeof(ST\_SIGNALING\_SEND\_BUFFER));

        pSendBuffer->head.cmd = SINGNALING\_CHANNEL\_CMD\_LOGIN\_NEED\_SESSION;

        if (0 != m\_pTcpStream->SyncWriteAndRead(pSendBuffer, sizeof(signaling\_channel\_head\_t), &msg\_key\_recv, sizeof(signaling\_channel\_head\_t), 5))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- request key error -- 1");*

            break;

        }

*//*

        if ((msg\_key\_recv.cmd != SINGNALING\_CHANNEL\_CMD\_LOGIN\_NEED\_SESSION) || (msg\_key\_recv.result != SINGNALING\_CHANNEL\_RESULT\_OK))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- request key error -- 2");*

            break;

        }

*//*

        uint32\_t m\_nSessionID = msg\_key\_recv.session;

*//--------------------------------------------------------------------------------------------------------------------------*

*//使用key加密，登陆*

        signaling\_channel\_user\_login\_t stLogin;

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

        XDESEncode(m\_szUser, m\_nSessionID, stLogin.userdes);

        XDESEncode(m\_szPwd, m\_nSessionID, stLogin.pwddes);

*//*

        memset(pSendBuffer, 0, sizeof(ST\_SIGNALING\_SEND\_BUFFER));

        pSendBuffer->head.cmd = SINGNALING\_CHANNEL\_CMD\_LOGIN;

        pSendBuffer->head.session = m\_nSessionID;

        pSendBuffer->head.datalen = sizeof(stLogin);

*//*

        memcpy(pSendBuffer->msg, &stLogin, sizeof(stLogin));

*//*

        if (0 != m\_pTcpStream->SyncWriteAndRead(pSendBuffer, sizeof(signaling\_channel\_head\_t) + sizeof(stLogin), &msg\_key\_recv, sizeof(signaling\_channel\_head\_t)))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- login error -- 1");*

            break;

        }

        if ((msg\_key\_recv.cmd != SINGNALING\_CHANNEL\_CMD\_LOGIN) || (msg\_key\_recv.result != SINGNALING\_CHANNEL\_RESULT\_OK))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- login error -- 2");*

            break;

        }

*//--------------------------------------------------------------------------------------------------------------------------*

        pSendBuffer->head.cmd = SINGNALING\_CHANNEL\_CMD\_REDIRECTION\_CLASSIC\_GET\_DEVICES;

        pSendBuffer->head.datalen = 0;

        if (0 != m\_pTcpStream->SyncWriteAndRead(pSendBuffer, sizeof(signaling\_channel\_head\_t), &msg\_key\_recv, sizeof(signaling\_channel\_head\_t), 5))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- request key error -- 1");*

            break;

        }

        if ((msg\_key\_recv.cmd != SINGNALING\_CHANNEL\_CMD\_REDIRECTION\_CLASSIC\_GET\_DEVICES) || (msg\_key\_recv.result != SINGNALING\_CHANNEL\_RESULT\_OK))

        {

*//CROSS\_TRACE("CMediaDevInterface::Login -- request key error -- 2");*

            break;

        }

*//*

        if (msg\_key\_recv.datalen > 0)

        {

            if (0 != m\_pTcpStream->SyncWriteAndRead(NULL, 0, pdstBuffer, msg\_key\_recv.datalen, 5))

            {

*//CROSS\_TRACE("CMediaDevInterface::Login -- request key error -- 1");*

                break;

            }

        }

*//*

        ClearDeivceInfoList();

        for (**int** i = 0; i < (**int**)(msg\_key\_recv.datalen / sizeof(ST\_CLASSIC\_DEVICE\_INFO\_BASE)); i++)

        {

            ST\_CLASSIC\_DEVICE\_INFO\_BASE \* pDeviceInfo = CClassicDeviceInfoBufferPool::Instance().malloc();

            memcpy(pDeviceInfo, pdstBuffer + i\*sizeof(ST\_CLASSIC\_DEVICE\_INFO\_BASE), sizeof(ST\_CLASSIC\_DEVICE\_INFO\_BASE));

            m\_lisDevice[pDeviceInfo->dev\_id] = pDeviceInfo;

        }

        UpdateSignalingServers();

*//*

        len = msg\_key\_recv.datalen;

    } while (0);

    if (m\_pTcpStream)

    {

        m\_pTcpStream->Release();

        m\_pTcpStream = NULL;

    }

    CSignalingBufferPool::Instance().free(pSendBuffer);

    m\_cs.Unlock();

}