

Network_Info

This article gives you an idea/wrapper class for getting Network information like Current MCC/MNC/LAC, Home MCC/MNC/LAC, GPRS status, Device IP, IMSI. The class is easy to use.

Header File

```
#ifndef __NW_INTERFACE_2ND_ED_H__
#define __NW_INTERFACE_2ND_ED_H__

#include <etel.h>
#include <etelmm.h>

typedef struct _NetworkInfo
{
    int mcc;    /*!< - Mobile country code */
    int mnc;    /*!< - Mobile network code */
    int lac;    /*!< - Location area code */
} NetworkInfo;

class CNWInterf2ndEd : public CBase
{
public:
    static CNWInterf2ndEd* NewL();
    static CNWInterf2ndEd* NewLC();
    ~CNWInterf2ndEd();
    TInt GetIMSI(char* aImsiNum, int aSize);
    TInt GetCurrentNWInfo( NetworkInfo& aNetworkInfo );
    TInt GetHomeNWInfo( NetworkInfo& aNetworkInfo );
    TInt GetDeviceIp(char* aPhoneIp, int aSize);
    TInt GetGprsStatus();

private:
    CNWInterf2ndEd();
    void ConstructL();
    RTelServer  iTelserver;
    RMobilePhone  iPhone;
    RLine  iLine;
    TBuf <128>  iPhoneNumber;
    TBuf <128>  iLineName;
    TInt  iNumberPhones;
    RTelServer::TPhoneInfo  iPhoneInfo;
    RPhone::TLineInfo  iLineInfo;
    TRequestStatus  iNWSstatus;
};

#endif //__NW_INTERFACE_2ND_ED_H__
```

Source File

```
#include <string.h>
#include <in_sock.h>

CNWInterf2ndEd* CNWInterf2ndEd::NewL()
```

Network_Info

```
{
    CNWInterf2ndEd CNWInterf2ndEd::NewLC();
    CleanupStack;
return self;
}

CNWInterf2ndEd* CNWInterf2ndEd::NewLC()
{
    CNWInterf2ndEd new (ELeave) CNWInterf2ndEd;
    CleanupStack(self);
    ~ConstructL();
return self;
}

CNWInterf2ndEd::~CNWInterf2ndEd()
{
    Close();
    Close();
    iTelServer;
}

CNWInterf2ndEd::CNWInterf2ndEd()
{
}

void CNWInterf2ndEd::ConstructL()
{
    User::LeaveIfError(iTelServer.Connect(RTelServer::KDefaultMessageSlots));
    _LIT (KTsyName, "phonetsy.tsy");
    User::LeaveIfError(iTelServer.LoadPhoneModule(KTsyName));
    //Find the number of phones available from the tel server
    User::LeaveIfError(iTelServer.EnumeratePhones(iNumberPhones));
    //Check there are available phones
    if ( iNumberPhones < 1 )
    {
        User::Leave(KErrNotFound);
    }
    //Get info about the first available phone
    User::LeaveIfError(iTelServer.GetPhoneInfo(0, iPhoneInfo));
    iPhoneName.Copy(iPhoneInfo.iName);
    //Use this info to open a connection to the phone, the phone is identified by itsname
    User::LeaveIfError(iPhone.Open(iTelServer, iPhoneName));
    //Get info about the first line from the phone
    User::LeaveIfError(iPhone.GetLineInfo(0, iLineInfo));
    iLineName.Copy(iLineInfo.iName);
    //Use this line to open a line
    User::LeaveIfError(iLine.Open(iPhone, iLineName));
}

TInt CNWInterf2ndEd::GetIMSI(char* aImsiNum, int aSize)
{
    TInt ret = KErrGeneral;
    TRequestStatus lSIMStatus ;
    RMobilePhone::TMobilePhoneSubscriberId lSubscriberIdentity;
    iPhone.GetSubscriberId(lSIMStatus, lSubscriberIdentity);
    User::WaitForRequest(lSIMStatus);
    if ( lSIMStatus == KErrNone )
    {
        TBuf8<50> imsi;
        imsi.FillZ(50);
        imsi.Copy(lSubscriberIdentity);
        memset(aImsiNum, '\0', aSize);
    }
}
```

Network_Info

```
        strncpy(aImsiNum, (const char *)imsi.Ptr(), aSize);
        ret = KErrNone;
    }
    return ret;
}

TInt CNWInterf2ndEd::GetCurrentNWInfo( NetworkInfo& aNetworkInfo )
{
    TInt ret = KErrGeneral;
    TRequestStatus lNetworkStatus ;
    RMobilePhone::TMobilePhoneNetworkInfoV1 infov1;
    RMobilePhone::TMobilePhoneNetworkInfoV1Pckg statusPkg(infov1);
    RMobilePhone::TMobilePhoneLocationAreaV1 locArea;
    iPhone.GetCurrentNetwork(lNetworkStatus, statusPkg, locArea);
    User::WaitForRequest(lNetworkStatus);
    if ( lNetworkStatus == KErrNone )
    {
        TBuf8<50> country;
        TBuf8<50> network;
        country.Copy(infov1.iCountryCode);
        network.Copy(infov1.iNetworkId);
        aNetworkInfo.mcc=atoi((const char *)country.Ptr());
        aNetworkInfo.mnc=atoi((const char *)network.Ptr());
        ret = KErrNone;
    }
    else
    {
        // error case
    }
    return ret;
}

TInt CNWInterf2ndEd::GetGprsStatus()
{
    TInt ret = KErrGeneral;
    RSystemAgent iIndicatorSystemAgent
    iIndicatorSystemAgent();
    int availability=iIndicatorSystemAgent.GetState(KUidGprsAvailability);
    int status=iIndicatorSystemAgent.GetState(KUidGprsStatus);
    if(availability==ESAGprsAvailable && (status==ESAGprsAttach || status==ESAGprsContextActive))
    {
        =1; ret
    }
    else
    {
        =0; ret
    }
    return ret;
}

TInt CNWInterf2ndEd::GetHomeNWInfo( NetworkInfo& aNetworkInfo )
{
    TInt ret = KErrGeneral;
    TRequestStatus lNetworkStatus ;
    RMobilePhone::TMobilePhoneNetworkInfoV1 infov1;
    RMobilePhone::TMobilePhoneNetworkInfoV1Pckg statusPkg(infov1);
    iPhone.GetHomeNetwork(lNetworkStatus, statusPkg);
    User::WaitForRequest(lNetworkStatus);
    if ( lNetworkStatus == KErrNone )
    {
        TBuf8<50> country;
        TBuf8<50> network;
```

Network_Info

```
country.Copy(infov1.iCountryCode);
network.Copy(infov1.iNetworkId);
aNetworkInfo.mcc=atoi((const char *)country.Ptr());
aNetworkInfo.mnc=atoi((const char *)network.Ptr());
ret = KErrNone;
}
else
{
// error case
}
return ret;
}

TInt CNWInterf2ndEd::GetDeviceIp(char* aPhoneIp, int aSize)
{
TRequestStatus status;
RSocketServ lmSocketServ;
RSocket lmSocket;
RHostResolver lmHostResolver;
TNameEntry lmNameEntry;
TNameRecord lmNameRecord;
// Open channel to Socket Server
User::LeaveIfError(lmSocketServ.Connect());

// Open a TCP socket
User::LeaveIfError(lmSocket.Open(lmSocketServ, KAfInet, KSockDatagram, KProtocolInetUdp));

// Initiate DNS
User::LeaveIfError(lmHostResolver.Open(lmSocketServ, KAfInet, KProtocolInetUdp));
TBuf<100> hostname;
lmHostResolver.GetHostName(hostname, status);
User::WaitForRequest(status);
lmHostResolver.GetByName(hostname, lmNameEntry, status);
User::WaitForRequest(status);
lmNameRecord = lmNameEntry();

/* If the family is KAfInet6, the output buffer must be at least 39 characters.
If less, the buffer is filled with '*' characters.*/

TBuf<50> ipAddr;
TInetAddr::Cast(lmNameRecord.iAddr).Output(ipAddr);
char* ipAddress = CKnAvsAppUiUtils::EncodeUtf7ExtractCharL(ipAddr);
strncpy(aPhoneIp, (const char *)ipAddress, aSize);
delete ipAddress;
lmHostResolver.Close();
lmSocket.Close();
lmSocketServ.Close();
return KErrNone;
}
```