

Contents

- [1 Access point \(IAP\) selection dialog](#)
- [2 Reading all access points from the device](#)
- [3 Reading GPRS/CSD internet access points from the device](#)
- [4 Retrieving Proxy Server Name from IAP](#)

Access point (IAP) selection dialog

Reviewer Approved



UI Libraries used:

```
LIBRARY avkon.lib
LIBRARY eikcoctl.lib
```

Add this to your header file:

```
class TIapData
{
public:
    TBuf<128> iName;
    TUint32 iIap;

};
```

These are the additional headers files required:

```
#include <aknlists.h> // CAknSinglePopupMenuStyleListBox
#include <CommDbConnPref.h>
#include <CommDb.h> // CCommsDatabase

TInt SelectIAPL()
{
    CArrayFixFlat<TIapData>* iEApList=new (ELeave) CArrayFixFlat<TIapData>(2) ;
    TInt i;
    // Make listitems. and PUSH it
    CAknSinglePopupMenuStyleListBox new(ELeave) CAknSinglePopupMenuStyleListBox;
    CleanupStackPush(i);stack++;

    // Create popup list and PUSH it.
    CAknPopupMenuList = CAknPopupMenuList::NewL(list,
        R_AVKON_SOFTKEYS_OK_CANCEL, :AKMOPopupDown);
    CleanupStackPush(i);stack++;

    CDesCArrayFlat* iEApList= new (ELeave) CDesCArrayFlat(5);
    CleanupStackPush(i);stack++;
    // initialize listbox.
    ->ConstructL(popupList, CEikListBox::ELeftDownInViewRect);
    ->CreateScrollBarFrameL(ETrue);
    ->ScrollBarFrame()->SetScrollBarVisibilityL(CEikScrollBarFrame::EOff,
        CEikScrollBarFrame);

    <52>TIapListfromtable;
```

Reading_internet_access_points_from_the_device

```

TferrNone;

CCommsDatabaseDB=CCommsDatabase::NewL(EDatabaseTypeIAP);
CleanupStack(iCommsDB);stack++;
#ifdef __SERIES60_3X__
CCommsDbTableViewTable = iCommsDB->OpenIAPTableViewMatchingBearerSetLC(
    ECommDbBearerPFSAN|ECommDbBearerVirtual,
    ECommDbConnectionDirectionOutgoing
#else
CCommsDbTableViewTable = iCommsDB->OpenIAPTableViewMatchingBearerSetLC(
    ECommDbBearerPFSAN,
    ECommDbConnectionDirectionOutgoing
#endif
::LeaveIfError(gprsTable->GotoFirstRecord());
=0;Int i
TUint32 id
TIapData eap

TferrNone; //current value
do
{
    ->ReadTableC(COMMDB_NAME), iapfromtable);
    ->ReadTableC(COMMDB_ID), id);
    ->AppendL(iapfromtable);
    iIap = id; eap.
    iName.Copy(iapfromtable);
    ->AppendL(iap);

    = gprsTable->GotoNextRecord();
    ++; i
}
while (err == KErrNone);
CleanupStackDestroy(2); stack--;

// Set listitems.
CTextListModel= list->Model();
->SetTextArray(items);
->SetOwnershipType(ELbmOwnsItemArray);
CleanupStack

popupListL(_L("IAP"));
->SetListBoxObserver(popupList);
TInt popupOk=popupList->ExecuteLD();
CleanupStack
TInt iap
if (popupOk)
{
    = list->GetCurrentItemIndex();
    =(*iEApList)[iapex].iIap;
}

CleanupStackDestroy();
iEApList);
delete iEApList;
return iap;
}

```

Reading all access points from the device

The following code reads all access points from the device disregarding their type. This is preferred unless there is some grave reason why you would force the user to select e.g. from GPRS connections.

```
TFileName iapName;
TUint32 iapID;
TInt err;

// open the IAP communications database
CCommsDatabase* commDB = CCommsDatabase::NewL();
CleanupStack::PushL(commDB);

// Open the IAP table
CCommsDbTableView* view = commDB->OpenTableLC(TPtrC(IAP));

// Point to the first entry
if (view->GotoFirstRecord() == KErrNone)
{
do
{
    ->ReadTextL(VEwC(COMMDB_NAME), iapName);
    ->ReadUintL(VEwC(COMMDB_ID), iapID);

// Store name and ID to where you want to
} while (err = view->GotoNextRecord(), err == KErrNone);
}

CleanupStack::PopAndDestroy(); // view
CleanupStack::PopAndDestroy(); // commDB
```

Reading GPRS/CSD internet access points from the device

Following code sample shows how to retrieve all set GPRS or CSD internet access point from the device.

```
CCommsDatabase* commDb = CCommsDatabase::NewL(EDatabaseTypeIAP);
CleanupStack::PushL(commDb);

CCommsDbTableView* commView = commDb->OpenIAPTableViewMatchingBearerSetLC(
    ECommDbBearerCSD|ECommDbBearerGPRS, ECommDbConnectionDirectionOutgoing);

TFileName TmpName;

if (commView->GotoFirstRecord() == KErrNone)
{
do
{
    Zero(); TmpName.
    ->ReadTextL(VEwC(COMMDB_NAME), TmpName);
```

Reading_internet_access_points_from_the_device

```
// Add TmpName to CDesCArray for example
// it has the IAP's human readable name

        ; TUInt32 IapNum
        ->ReadUIntL(TPtrC(COMMDB_ID), IapNum);

// Add IapNum into other array,
// it has the ID for the accesspoint used with connection

}while (commView->GotoNextRecord() == KErrNone);
}

CleanupStack::PopAndDestroy(2); //commView, commDb
```

With the code the TmpName variable will hold the internet access point name you could show to user for selection, and the IapNum variable holds the internet access point identifier you could then use with the RConnection when making a silent connection. Note that when making a silent connection you also need to call the SetDialogPreference()-function with ECommDbDialogPrefDoNotPrompt for the connection preferences (TCommDbConnPref).

Retrieving Proxy Server Name from IAP

The code below shows how to retrieve proxy server name and the port number from a certain IAP.

```
//
// The tableView variable here is the instance of CCommsDbTableView
// Normally you call CCommsDatabase::OpenTableLC() to get the instance of
// CCommsDbTableView.
// (See also the other examples above.)
//

// Firstly, we need to get the IAPService and IAPServiceType
// of our IAP in order to find out the proxy information.
TUInt32 iapService;
HBufC* iapServiceType;
tableView->ReadUIntL(TPtrC(IAP_SERVICE), iapService);
iapServiceType = tableView->ReadLongTextLC(TPtrC(IAP_SERVICE_TYPE));

// Check whether this IAP uses proxy or not.
TBool isProxyEnabled = EFalse;
CCommsDbTableView* proxyTableView = commsDb->OpenViewOnProxyRecordLC(
    iapService, *iapServiceType);
if (KErrNone == proxyTableView->GotoFirstRecord())
{
    proxyTableView->ReadBoolL(TPtrC(PROXY_USE_PROXY_SERVER), isProxyEnabled);

    // If proxy is enabled then do something.
    if (isProxyEnabled)
    {
        proxyServerName = proxyTableView->ReadLongTextLC(TPtrC(PROXY_SERVER_NAME));
        proxyTableView->ReadUIntL(TPtrC(PROXY_PORT_NUMBER), proxyPortNumber);

        // Do whatever you want with proxyServerName and proxyPortNumber.
    }
}
```

Reading_internet_access_points_from_the_device

```
CleanupStack::PopAndDestroy(proxyServerName);  
}  
}
```