Archived:How to add contacts from phonebook

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Warning: API is deprecated in S60 3.2. Consider this while using in later versions of Symbian

There are plenty of occasions when you need to add contacts from Phonebook to your Symbian C++ application. Examples are SMS, Call Control applications etc. To do that you have to first load the phonebook's resource file like

```
RPbkViewResourceFile phonebookResource( *(CEikonEnv::Static()) );
phonebookResource.OpenL();
```

In general you would want to allow users to fetch multiple entries from the phonebook dialog so you can use `CPbkMultipleEntryFetchDlg` and pass the correct parameters. Also attach the correct Phonebook Contact Engine to the dialog. (Here I am picking the default contact book, whereas you may want to display your own special contact book database.)

The code may look like

```
CPbkMultipleEntryFetchDlg::TParams params;
CleanupStack::PushL(params);

CPbkContactEngine* iPbkContactEngine =
   CPbkContactEngine::NewL(&iEikonEnv->FsSession());
CleanupStack::PushL(iPbkContactEngine);
params.iContactView = &iPbkContactEngine->AllContactsView();

CPbkMultipleEntryFetchDlg* fetcher =
   CPbkMultipleEntryFetchDlg::NewL(params, *iPbkContactEngine);
fetcher->SetMopParent(this);
```

Later just execute the dialog. On return, the dialog will give you a list of contact ids with which you can access individually selected contacts from Contact Engine and process them

```
TBuf<30> phoneNumber;

TInt paramCount = params.iMarkedEntries->Count();
// Get the selected contacts id array
for ( TInt i = 0; i < paramCount; ++i )
{
 const TContactItemId cid = ( *params.iMarkedEntries )[i];
 // Open the selected contact using Phonebook engine,
 // choose correct number (launch list query if needed)
 CPbkContactItem* pbkItem = iPbkContactEngine->ReadContactLC( cid );
 TPbkContactItemField* tmp;
 if ((tmp = pbkItem->FindField(EPbkFieldIdPhoneNumberMobile)) != NULL)
 {
  phoneNumber = tmp->Text();
  if(phoneNumber.Length() > 0){
   if(iData)
                      iAppView->AddNumber(phoneNumber);
  }
 }

        CleanupStack::PopAndDestroy(1);//CPbkContactItem
}
```

If you want to implement the behavior that, if a contact item has more than one number, and you want to show native dialog to choose between the numbers(like in native Sms application), you can use following code

```
TBuf<30> phoneNumber;

TInt paramCount = params.iMarkedEntries->Count();
// Get the selected contacts id array
for ( TInt i = 0; i < paramCount; ++i )
{
 TBuf<128> phoneNumber;
 TBuf<128> phoneName;
 TBool namefound = EFalse;
 const TContactItemId cid = ( *params.iMarkedEntries )[i];
 // Open the selected contact using Phonebook engine,
 // choose correct number (launch list query if needed)
 CPbkContactItem* pbkItem = iPbkContactEngine->ReadContactLC( cid );
        //Using CPbkSmsAddressSelect dialog for user to choose between numbers
 CPbkSmsAddressSelect* dlg = new (ELeave) CPbkSmsAddressSelect();
 CPbkSmsAddressSelect::TParams addparams(*pbkItem);
 TBool selected = dlg->ExecuteLD(addparams);
 //If selected is ETrue, it means that user has selected some number, else user has pressed cancel
 //Also, if contact item has only one number, then there is no popup shown to the user and returned value is ETrue
 if(selected){
                //Getting the selected field, i.e. Mobile, Mobile (Business),Tel, Tel. (Home)
                //You can find the exact label of field by numtmp->Label()
  const TPbkContactItemField* numtmp = addparams.SelectedField();
  if(numtmp){
   phoneNumber = numtmp->Text();
  }
  TPbkContactItemField* nametmp;
  if ((nametmp = pbkItem->FindField(EPbkFieldIdFirstName)) != NULL)
  {
   phoneName = nametmp->Text();
   if(phoneName.Length() > 0){
```


(C) Copyright Nokia 2013. All rights reserved.

```
      namefound = ETrue;
    }
   }
   if ((nametmp = pbkItem->FindField(EPbkFieldIdLastName)) != NULL)
   {
    if(namefound){
            phoneName.Append(_L(" "));
     phoneName.Append(nametmp->Text());
    }
    else{
      phoneName = nametmp->Text();
    }
   }
  }
        CleanupStack::PopAndDestroy(1);//CPbkContactItem
 }
```

In the end clean up your stack and close phonebook resource

```
//CPbkContactEngine, CPbkMultipleEntryFetchDlg::TParams
CleanupStack::PopAndDestroy(2);
phonebookResource.Close();
```

Headers required

```
#include <rpbkviewresourcefile.h>
#include <cpbkmultipleentryfetchdlg.h>
#include <cpbkcontactengine.h>
#include <cpbkcontactitem.h>
#include <cpbksmsaddressselect.h>
```

Library needed

```
LIBRARY pbkview.lib pbkeng.lib
```

Capability required

```
Capability ReadDeviceData ReadUserData WriteDeviceData WriteUserData
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.