# Changing application's language at runtime

**Changing application's language at runtime.**

There're many examples that show how to write multi-language applications. The basic idea is getting the proper language at installation time, or through the use of optional packs. Sometimes, there's a need to change languages dynamically while the application is running. This may seem like a difficult task, but following some basic guidelines it'll prove to be very easy.

First off, we'll need to separate the default resource file into two pieces. Depending on the way you choose to allow the user to toggle among languages, you'll provide the default resource file of the corresponding structures. For example purposes, I've chosen a listbox dialog, which is the first control shown upon execution. Depending on the language selected, the corresponding secondary resource file will be loaded. This second resource file will be the provider of the localized content. So back to our example, the default resource file will look like:

```
NAME TEST

#include <eikon.rh>
#include <avkon.rsg>
#include <avkon.rh>
#include <avkon.mbg>

#include "Test.hrh"
#include "Test.loc"

RESOURCE RSS_SIGNATURE { }
RESOURCE TBUF { buf = qtn_app_caption_string; }
RESOURCE EIK_APP_INFO
{
    cba = R_AVKON_SOFTKEYS_OPTIONS_BACK;
    menubar = r_dummy_menubar;
}

RESOURCE MENU_BAR r_dummy_menubar
{
    titles=
    {
        MENU_TITLE { menu_pane = r_dummy_menupane; txt=""; }
    };
}

RESOURCE MENU_PANE r_dummy_menupane
{
    items=
    {
        MENU_ITEM { command = EAknCmdExit; txt = qtn_cmd_exit; }
    };
}

RESOURCE DIALOG r_list_dialog
{
    flags = EAknDialogSelectionList | EEikDialogFlagWait;
    buttons = R_AVKON_SOFTKEYS_SELECT_CANCEL;

    items =
    {
        DLG_LINE
        {
            type = EAknCtSingleListBox;
            id = ESelectionListControl;
            control = LISTBOX
            {
                flags = EAknListBoxSelectionList;
            };
        }
    };
}
```

First thing to note is the use of cba and menubar in the EIK_APP_INFO resource structure. This is a convenience, as it delegates construction and layout of both CBA and menu bar to the application's UI. Otherwise, we'd have to perform this action manually (which means calculating the screen area and location for each of these controls, among other things) which is also specific to the UI used (Avkon, Qikon, etc). The last resource structure defined corresponds to the dialog that will allow us to select the language dynamically.

Now we define a second resource file, which will contain the rest of the resources used by the application. Most controls should be in this resource file:

```
NAME EXTR

#include <eikon.rh>
#include <avkon.rsg>
#include <avkon.rh>
#include <avkon.mbg>

#include "Test.hrh"
#include "Test.loc"

RESOURCE RSS_SIGNATURE { }

RESOURCE CBA r_softkeys
{
    buttons =
    {
        AVKON_CBA_BUTTON { id = EAknSoftkeyOptions; txt = qtn_softkey_options; },
        AVKON_CBA_BUTTON { id = EAknSoftkeyExit; txt = qtn_softkey_exit; }
    };
}

RESOURCE MENU_BAR r_menubar
{
    titles=
    {
        MENU_TITLE { menu_pane = r_menupane; txt=""; }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    };
}

RESOURCE MENU_PANE r_menupane
{
    items=
    {
        MENU_ITEM { command = ECmdLang; txt = qtn_cmd_lang; },
        MENU_ITEM { command = ECmdTest; txt = qtn_cmd_test; },
        MENU_ITEM { command = EAknCmdExit; txt = qtn_cmd_exit; }
    };
}

RESOURCE TBUF r_text_greeting
{
    buf = qtn_text_greeting;
}
```

Also note we need to provide a NAME and RSS_SIGNATURE, otherwise we'd get a panic (internally, CONE's AddResourceFileL() uses RResourceFile and RResourceFile::ConfirmSignatureL() to verify the signature and initialize the offset)

And this is the function that will scan the available localized resource files. For simplicity, the list dialog shows the corresponding file names. You may want to provide some mapping to more user-friendly names. Also, for clarity's sake, only basic error handling is done.

```
void CTestAppUi::ChooseLanguageL()
{
    HBufC* appName = Application()->AppFullName().AllocLC();
    TParsePtrC parse(*appName);

    // Construct search path
    _LIT(KExtra, "Extra.r??");
    TFileName file;
    file.Copy(parse.DriveAndPath());
    file.Append(KExtra);

    RFs& fs = iCoeEnv->FsSession();

    CDir* entries;
    fs.GetDir(file, KEntryAttNormal, ESortByName, entries);
    CleanupStack::PushL(entries);

    CDesCArray* list = new(ELeave) CDesCArrayFlat(5);
    CleanupStack::PushL(list);

    // Fill up listbox
    for (TInt i = 0; i < entries->Count(); ++i)
    {
        _LIT(KFormat, "\t%S");
        file.Format(KFormat, &(*entries)[i].iName);
        list->AppendL(file);
    }

    TInt index;
    CAknSelectionListDialog* dialog = CAknSelectionListDialog::NewL(index, list, 0);
    if (dialog->ExecuteLD(R_LIST_DIALOG))
    {
        file.Copy(parse.DriveAndPath());
        file.Append((*entries)[index].iName);

        // Replace resource file
        if (iOffset)
            iCoeEnv->DeleteResourceFile(iOffset);

        iOffset = iCoeEnv->AddResourceFileL(file);

        // Load cba, menu, etc as necessary
        iEikonEnv->AppUiFactory()->Cba()->SetCommandSetL(R_SOFTKEYS);
        iEikonEnv->AppUiFactory()->MenuBar()->SetMenuTitleResourceId(R_MENUBAR);
    }

    CleanupStack::PopAndDestroy(3, appName);    // list, entries, appName
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.