Series 60 Platform

AVKON Common File Dialogs Usage

Glossary

Abbreviation/Term	Definition
CFD	Common File Dialogs
DLL	Dynamic Link Library
AVKON	Use Interface classes

1. Usage

1.1 Project's Definition File (.MMP)

Include header files of AVKON Common Dialogs to your project by adding the following line in the .MMP file:

```
SYSTEMINCLUDE \epoc32\include
```

Also link the project against library CommonDialogs.LIB with the following line of code:

```
LIBRARY CommonDialogs.lib
```

1.2 The Resource File (.RSS)

Dialogs can be created from a resource or with function parameters. This chapter defines the resources that can be used for creating dialogs.

When defining resources, add the following to the beginning of your resource file:

```
#include <CommonDialogs.hrh> // Enumerations
#include <CommonDialogs.rh> // Resource structures
```

1.2.1 MEMORYSELECTIONDIALOG resource

This resource is for defining a memory selection dialog. The resource structure looks like this:

```
STRUCT MEMORYSELECTIONDIALOG
           {
           LTEXT title;
           LTEXT softkey_1 = text_softkey_ok;
           LTEXT softkey_2 = text_softkey_cancel;
           STRUCT locations[]; // LOCATION
           }
```

The first item, “title”, defines the title to be used with memory selection dialog. “softkey_1” defines the text that is used for left softkey in control_pane_1. “softkey_2” defines text for right softkey. Default values for them are “Ok” and “Cancel”. The next field “locations” defines paths for memories. LOCATION structure looks like this:

```
STRUCT LOCATION
           {
           LTEXT root_path;
           LTEXT default_folder;
           }
```

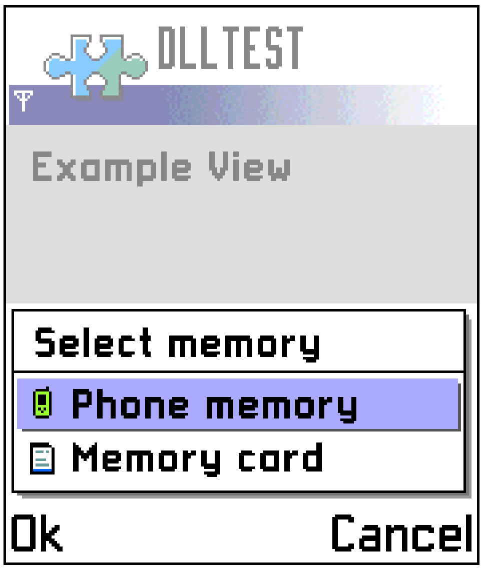
It defines a root path and a default folder. "root_path" is the absolute path to be considered as root when browsing, e.g. "C:\Nokia\". "default_folder" is the folder where browsing is started, e.g. "Images\Pictures\". LOCATION structures must be defined for phone memory and memory card.

An example memory selection resource could look like this:

```
RESOURCE MEMORYSELECTIONDIALOG r_memory_selection_dialog
           {
           title = "Choose memory:";
           softkey_1 = "Accept";
           softkey_2 = "Negative";
           locations =
                       {
                       LOCATION { root_path = "C:\\Nokia\\"; },
                       LOCATION { root_path = "E:\\"; default_folder =
"Images\\"; }
                       };
           }
```

In this example, root path for phone memory is “C:\Nokia\” (remember the trailing backslash) and browsing is started from that directory if phone memory is selected. Root path for memory card is “E:\” and the browsing is started from the default folder, “E:\Images\”. You can leave any of the resource items undefined. In that case, default values are used.

The memory selection dialog looks like this:



1.2.2 FILESELECTIONDIALOG resource

This resource is for defining a file selection dialog. The resource structure looks like this:

```
STRUCT FILESELECTIONDIALOG
           {
           LTEXT title;
           LTEXT softkey_1_file = text_softkey_select;
           LTEXT softkey_1_folder = text_softkey_open;
           LTEXT softkey_2_root_level = text_softkey_cancel;
           LTEXT softkey_2_subfolder = text_softkey_back;
           LTEXT root_path;
           LTEXT default_folder;
           STRUCT filters[]; // FILTER structs
           }
```

Titles and softkeys can be defined similarly to the memory selection dialog. “softkey_1_file” is shown when a file is focused and “softkey_1_folder” is shown when a folder is focused. “softkey_2_root_level” is shown when user is browsing the root folder and “softkey_2_subfolder” when user is browsing under a subdirectory. Root path and default folder must be defined if file selection dialog is launched directly without memory selection dialog. Filters can be defined as an array of FILTER structures. Filter structure looks like this:

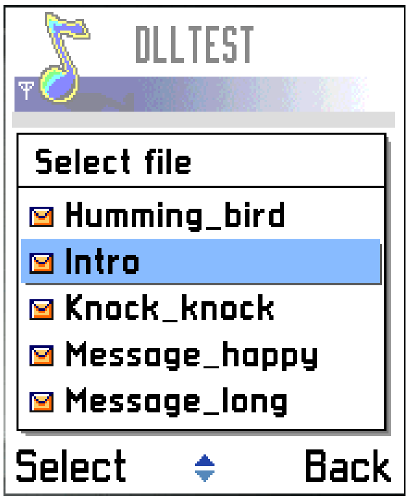
```
STRUCT FILTER
           {
           WORD filter_type;
           WORD filter_style;
           LTEXT filter_data[];
           }
```

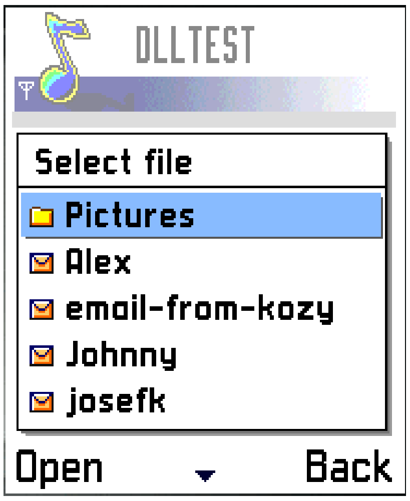
"filter_type" defines if the filter is an attribute or a filename filter. The enumerations are defined in CommonDialogs.hrh in TFilterType. "filter_style" defines if the filter is an inclusive or an exclusive filter. The enumerations are defined in CommonDialogs.hrh in TFilterStyle. "filter_data" is an array of strings. In filename filter, data are wildcards ("*.jpg", "*.gif”, ...). In attribute filter, data are first letters of attributes (“R”, “H”, “S”, “V”, “D”, “A” meaning “Read only”, “Hidden”, “System”, “Volume”, “Directory”, “Archive”). Note that CFD automatically filters out files with Hidden and System attributes from the list. You can leave any of the resource items undefined. In this case, default values are used.

This is an example of a file selection dialog resource:

```
RESOURCE FILESELECTIONDIALOG r_file_selection_dialog
           {
           title = "Select-a-file:";
           root_path = "C:\\Nokia\\Images\\";
           filters =
                       {
                       FILTER
                                   {
                                   filtertype = EAttributeFilter;
                                   filterstyle = EExclusiveFilter;
                                   filterdata = { "SH", “R” }; // Excludes
system, hidden and read-only attributes
                                   }
                       };
           }
```

The file selection dialog looks like this:





1.2.3 FILENAMEPROMPTDIALOG resource

This resource is for defining a filename prompt dialog. The resource structure looks like this:

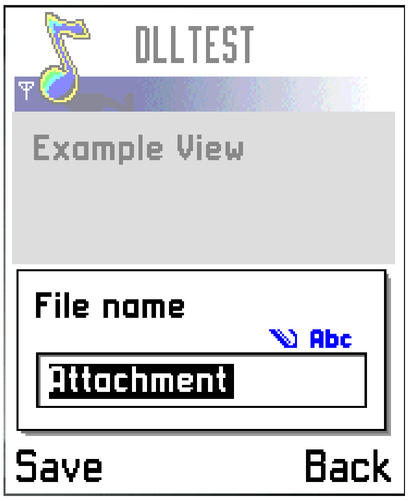
```
STRUCT FILENAMEPROMPTDIALOG
           {
           LTEXT filename_prompt; // title
           LTEXT default_filename;
           LTEXT path;
           LTEXT softkey_1 = text_softkey_ok;
           LTEXT softkey_2 = text_softkey_cancel;
           }
```

“filename_prompt” is the title for filename prompt dialog as “title” is the title for memory and file selection dialog. Softkeys are defined similarly to memory selection dialog. "default_filename" defines the text that is used as the default filename. This can consist of file name and extension. Extension is not shown to the user. “path” is the absolute path to where the file is to be saved. If not set, existence of similarly named files cannot be checked. You can leave any of the resource items undefined. In this case, default values are used.

An example resource could look like this:

```
RESOURCE FILENAMEPROMPTDIALOG r_filename_prompt_dialog
           {
           filename_prompt = "Enter filename:";
           default_filename = "Attachment.jpg";
           path = “C:\\Nokia\\Images\\”;
           }
```

A filename prompt dialog looks like this:



1.3 Application Source (.CPP)

All UI components (CAknFileNamePromptDialog, CAknFileSelectionDialog, CAknMemorySelection and CAknMemorySelectionSettingPage) can be created with NewL and launched with ExecuteL. There are also easier ways to launch dialogs: single function calls (RunDlgLD). Include headers of Common Dialogs that you need in your .CPP file:

```
#include <AknCommonDialogs.h> // For single function calls
#include <CAknMemorySelectionDialog.h>
#include <CAknMemorySelectionSettingPage.h>
#include <CAknFileSelectionDialog.h>
#include <CAknFileNamePromptDialog.h>
```

Example of launching a dialog:

```
_LIT(KDefaultFileName, "Attachment.jpg");
TFileName defaultFileName(KDefaultFileName);
AknCommonDialogs::RunSaveDlgLD(defaultFileName, R_MEMORY_SELECTION_DIALOG);
```

The resource:

```
RESOURCE MEMORYSELECTIONDIALOG r_memory_selection_dialog
{
locations =
           {
           LOCATION { root_path = "C:\\Nokia\\Images\\"; },
           LOCATION { root_path = "E:\\"; default_folder = "Images\\"; }
           };
}
```

More ways to launch Common Dialogs can be found by examining public headers and from the source code of DLLTEST application.