

ID	TSS000419	Creation date	September 21, 2006
Platform	S60 3rd Edition	Devices	
Category	Symbian C++	Subcategory	General Application Framework

Keywords (APIs, classes, methods, functions):

Overview

S60 MIME recognizers and opening files for editing

Description

It is a common requirement that developers would want to launch their custom file editors and viewers automatically when the "OK" key is pressed in the file browser. It is possible to achieve this using S60 file type recognizers. This solution describes how to write such a plug-in and also how to view and/or edit a file that has been opened.

Writing a recognizer

Recognizers on S60 3rd Edition need to provide a standard ECOM factory code to create an instance of the recognizer. All data recognizers must implement the polymorphic interface defined by `CApaDataRecognizerType`.

The pure virtual function `SupportedDataTypeL()` must be implemented. It returns the MIME type(s) that the recognizer is capable of recognizing.

For example, the code snippet can be as follows in your recognizer :

```
_LIT8(KExampleTextMimeType, "text/example");
TDataType CExampleRecognizer::SupportedDataTypeL(TInt /*aIndex*/) const
{
    return TDataType(KExampleTextMimeType);
}
```

All recognizers must implement also the `DoRecognizeL()` function that is called to do the data recognition. Implementations should set `iDataType` to the MIME type it considers the data to belong to and `iConfidence` to indicate a confidence rating for the recognition process.

The function can look like this in the recognizer :

```
void CExampleRecognizer::DoRecognizeL(const TDesC& aName, const TDesC8& aBuffer)
{
    _LIT8(KExampleData, "example");
    _LIT8(KDotExample, ".Example");
    TParse parse;
```

```

parse.Set(aName,NULL,NULL);
// extract the extension from the filename
TPtrC ext=parse.Ext();
if (ext.CompareF(KDotExample) == 0 &&
    aBuffer.FindF(KExampleData) != KErrNotFound)
    {
    iConfidence = ECertain;
    iDataType = TDataType(KExampleTextMimeType);
    }
}

```

The process of creating a recognizer is the same in S60 2nd Edition except that ECOM plugin framework is not used. In S60 2nd Edition, the recognizers are DLLs having .MDL extension and UID2 value of 0x10003A19. MDL recognizers are stored in the \system\recogs folder.

Registering for a MIME type

When developing an application to handle a particular MIME type, the application has to register itself giving the MIME type that it can handle. This is done in the client application's _reg.rss file.

The datatype_list section of the registration file lists the MIME types that the application supports, and the priority of support that each type is given. When a file is to be opened, Symbian OS launches the application that has the highest priority support for the data type.

An example of the file (Applications _reg.rss) is given below:

```

#include "HelloWorldBasic.loc"
#include <appinfo.rh>
#include <HelloWorldBasic.rsg>
UID2 KUidAppRegistrationResourceFile
UID3 0x10005B91
RESOURCE APP_REGISTRATION_INFO
{
    app_file="HelloWorldBasic";
    localisable_resource_file =
        qtn_helloworldbasic_loc_resource_file_1;
    localisable_resource_id =
        R_HELLOWORLDBASIC_LOCALISABLE_APP_INFO;
    embeddability = KAppEmbeddable;
    datatype_list =
        {
            DATATYPE
            {
                priority=EDatatypePriorityHigh;
                type="text/plain";
            }
        };
}

```

Passing file handle to viewer/editor

When a file is called by a system application like File Manager or Messaging App, the following function of the handler application is called:

```

CAknDocument::OpenFileL(
    CFileStore*& aFileStore,
    RFile& aFile)

```

where aFile is opened in read mode. So it's not possible for the client's application to make changes to the file directly. The solution for this is to open a new temporary file with the content of the file, edit the temporary file, and replace it with the original file.

Another way of doing this is to extract the filename and delegate the processing to a separate engine or the AppUi class:

```
void CHelloWorldBasicDocument::OpenFileL(
    CFileStore*& aFileStore, RFile& aFile)
{
    aFileStore=NULL;
    TFileName iFileName;
    aFile.FullName(iFileName);
    Process()->SetMainDocFileName(iFileName);
    TInt fileMode = EFileWrite|EFileShareExclusive;
    SetAppFileMode(fileMode);
    aFile.Close();
    iAppUi->LoadFile(iFileName);
    // File processing could be also delegated to
    // a separate engine or a view.
}
```

The difference between S60 2nd Edition and S60 3rd Edition is that in S60 2nd Edition,

```
CAknDocument::OpenFileL(
    TBool aDoOpen, const TDesc& aFileName, RFs& aFs)

```

gets called with the file name instead of handle. So, the problem with an already-opened file handle does not occur in S60 2nd Edition.

When the file that is being opened is in a private folder or there is no need to open it in write mode, then a duplicate of the file handle can be maintained for future use.

Opening a file in write mode

- Create a temporary file with the same contents as that of the original file. The contents can be extracted by using the RFs and RFile APIs and can be copied into the temporary file.
- Make the changes by either manipulating the temp. file directly or launching one's own view (similar to an editor).
- When saving, delete the original file and rename the temporary file with the original filename.

```
BafUtils::DeleteFile(iFsSession, iOriginalFile);
BafUtils::RenameFile(iFsSession, iTempFile, iOriginalFile);
```

Common problems:

One of the most common errors that one gets while trying to open a file in write mode while running in embedded mode is KErrInUse(-14).

This occurs because the Document Handler Framework opens the file in shareable read mode and shares the session with the application. Therefore, it's not possible for the application to open the file in write mode. In order to delete the original file and rename the temporary file, the handler application must wait for the OpenFileL() to return. It's not possible to delete or manipulate the file within OpenFileL().