

Contents

- [1 Mime Recogniser DLL \(.MDL\)](#)
- [2 MyRecognizer.h](#)
- [3 MyRecognizer.cpp](#)
- [4 MyRecognizer.mmp](#)

Mime Recogniser DLL (.MDL)

In S60, we often need to open a viewer application for files. To help Symbian OS to find the proper viewer for a file, application developers need to implement MIME-type recognisers, which map file extension or file content to a handler application. The resulting DLL has .MDL extension and it is installed in c:\system\recogs directory.

Implementing MDL

New MIME recognizers can be implemented by tuning existing ones. Here is a source code example for an imaginary mime type.

MyRecognizer.h

```
// includes
#include <apmrec.h> // For CApaDataRecognizerType
// Mime type string and the extension
_LIT8( KMyMimeType, "application/vnd.mymime" );
_LIT( KDotMymime, ".mym" );
// File header to look for from the data
_LIT8( KMyMimeHeader, "#!MY" );
// TUID of the recognizer
const TUid KUidMyMimeRecognizer( { 0x100530 } );
class CMyRecognizer : public CApaDataRecognizerType
{
public: // from CApaDataRecognizerType
CMyRecognizer();
virtual TUint PreferredBufSize();
virtual TDataType SupportedDataTypeL( TInt aIndex ) const;
private: // from CApaDataRecognizerType
virtual void DoRecognizeL(const TDesC& aName,
const TDesC& aBuffer );
// New funtions
private:
// Check the file name extension
TBool NameRecognized( const TDesC& aName );
// Look into the data
TBool HeaderRecognized( const TDesC& aName );
};
// End of file
```

MyRecognizer.cpp

```

// Includes
#include "MyRecognizer.h"
//
// Constructor
//
MyRecognizer::MyRecognizer()
:CApaDataRecognizerType(
KUidMyMimeRecognizer,
CApaDataRecognizerType::EHigh )
{
iCountDataTypes = 1;
}
//
// Preferred buffer size.
//
TUInt MyRecognizer::PreferredBufSize()
{
return 128;
}
//
// For the framework for collecting the supported mime types list.
//
TDataType MyRecognizer::SupportedDataTypeL( TInt aIndex ) const
{
switch( aIndex )
{
case 0:
default:
return TDataType( KMyMimeType );
break;
}
}
//
// The framework calls this function for recognition
//
void MyRecognizer::DoRecognizeL(
const TDesC& aName,
const TDesC8& aBuffer )
{
TBool nameOk( EFalse );
TBool headerOk( EFalse );
iConfidence = ENotRecognized;
if ( aBuffer.Length() < 10 )
return;
// First try the name, then the data.
nameOk = NameRecognized( aName );
headerOk = HeaderRecognized( aBuffer );
if ( nameOk && headerOk )
{
iConfidence = ECertain;
}
else if ( !nameOk && headerOk )
{
iConfidence = EProbable;
}
else if ( nameOk && !headerOk )
{
iConfidence = EPossible;
}
}

```

Implementing_recognisers

```
else
return;
iDataType = TDataType( KMyMimeType );
};
//
// Check if the file header can be recognized
//
TBool MyRecognizer::HeaderRecognized( const TDesC8& aBuffer )
{
if ( aBuffer.Find( KMyMimeHeader ) )
return ETrue;
else
return EFalse;
}
//
// Check if the file name has ".mym" extension
//
TBool MyRecognizer::NameRecognized( const TDesC& aName )
{
TBool ret = EFalse;
if ( aName.Length() > 5 )
{
TInt dotPos = aName.LocateReverse( '.' );
if (dotPos != KErrNotFound)
{
TInt extLength = aName.Length() - dotPos;
HBufC* ext = aName.Right( extLength ).AllocL();
CleanupStack::PushL( ext );
if ( ext->CompareF( KMyMimeHeader ) == 0 )
{
ret = ETrue;
}
CleanupStack::PopAndDestroy(); // ext
}
}
return ret;
}
//
// The gate function - ordinal 1
//
EXPORT_C CApaDataRecognizerType* CreateRecognizer()
{
CApaDataRecognizerType* thing = new MyRecognizer();
return thing; // NULL if new failed
}
//
// DLL entry point
//
GLDEF_C TInt E32Dll(TDllReason /*aReason*/)
{
return KErrNone;
}
// End of file
```

MyRecognizer.mmp

```
TARGET MyRecognizer.MDL
TARGETTYPE MDL
UID 0x10003A19 0x100530
TARGETPATH \system\recogs\
```

Implementing_recognisers

```
SOURCEPATH ..\src
SOURCE MyRecognizer.cpp
USERINCLUDE ..\inc
SYSTEMINCLUDE \epoc32\include
LIBRARY EUSER.LIB
LIBRARY APMIME.LIB
```