Implementing an ECOM interface

Overview

This code snippet demonstrates how to implement an ECom component from the base class.

Preconditions and important issues

- The UID2 (0x10009D8D) in the MMP file should be 0x10009D8D, which is common for all ECom plug-ins. The UID3 (0xE01F5465) is the ECom DLL UID.
- The UID of this interface can be selected from the protected or unprotected range, depending on the certification path you plan to follow. In this case we are using an unprotected UID so that the application can be self-signed.
- If the UID of the plug-in DLL is changed (and, correspondingly, the name of the rss file is changed), the ECom framework may not find the plug-in with the new UID. It may be necessary to delete the `ECom.ROM.Z.dat` and `Ecom.idx` files from the development environment folder of the SDK (for example, `C:\Symbian\9.1\S60_3rd_MR\Epoc32\winscw\c\private\10009D8F\`). When the emulator is restarted, the files are recreated.
- The ECom resource name must be identical to the ECom component/DLL name.
- Target type (`TARGETTYPE`) is PLUGIN.
- The compiled `EComExample.rsc` resource file must be copied to the WINSCW environment to `\Epoc32\release\winscw\udeb\z\resource\plugins`. See the `TARGETPATH` in the MMP file.

MMP file

```
TARGET EComExample.dll
TARGETTYPE PLUGIN

...

UID 0x10009D8D 0xE01F5465

...

SOURCE Proxy.cpp
SOURCE CMyHelloEcom.cpp  // Your ECom impl

...

// Import base class header and inline file
USERINCLUDE     ..\interface
SYSTEMINCLUDE   \epoc32\include
SYSTEMINCLUDE   \epoc32\include\ecom

// ECom resource name must be identical to ECom component/DLL name.
START RESOURCE E01F5465.rss
TARGET EComExample.rsc
TARGETPATH \resource\plugins
END
```

Header file

The header file is derived from the ECom base class.

```
// Include base class header
#include <CHelloEcomIF.h>
```


(C) Copyright Nokia 2014. All rights reserved.

```
class CMyHelloEcom : public CHelloEcomIF
    {
public:
    static CMyHelloEcom* NewL();
    virtual ~CImplementationClassOne();

    // Implementation of CHelloEcomIF
    void SayHello(TDes& aString);

private:
    CMyHelloEcom();
    void ConstructL();
    };

private:
    HBufC* iDescriptor;
```

Source file

The class must implement the virtual method `SayHello(TDes& aString)` defined in the base class.

```
#include "CMyHelloEcom.h"

CMyHelloEcom* CMyHelloEcom::NewL()
    {
    CMyHelloEcom* self=new(ELeave) CMyHelloEcom();
    CleanupStack::PushL(self);
    self->ConstructL();
    CleanupStack::Pop();
    return self;
    }

CMyHelloEcom::~CMyHelloEcom()
    {
    delete iDescriptor;
    }

CMyHelloEcom::CMyHelloEcom()
    {
    }

void CMyHelloEcom::ConstructL()
    {
    }

// Implementation of CHelloEcomIF
void CMyHelloEcom::SayHello(TDes& aString)
    {
    aString.Copy(*iDescriptor);
    }
```

Plug-in resource file

The resource file describes the contents of your plug-in DLL. It defines `IMPLEMENTATION_INFO` for individual implementations.

(C) Copyright Nokia 2014. All rights reserved.

`IMPLEMENTATION_INFO::implementation_uid` is used by ECOM framework's default resolver to find the requested implementation for a client.

```
#include "RegistryInfo.rh"

RESOURCE REGISTRY_INFO theInfo
    {
    // UID for the DLL
    dll_uid = 0xE01F5465;
    // Declare an array of interface info
    interfaces =
        {
        INTERFACE_INFO
            {
            // UID of the interface that is implemented
            interface_uid = 0xE0009DC1;
            implementations =
                {
                // Info for implementation of CHelloEcomIF
                IMPLEMENTATION_INFO
                    {
                    implementation_uid = 0xE0009DC7;
                    version_no = 1;
                    display_name = "ecomexample";
                    opaque_data = "";
                    default_data = "ecomexample";
                    }
                };
            }
        };
    }
```

Proxy

Defines where the implementations reside within the DLL. The ECom framework uses a chart of `TImplementationProxy` items to map implementation IDs to the locations of specific implementation instantiation methods (NewL). Each implementation ID within `TImplementationProxy` must match one in the resource file.

```
#include <e32std.h>
#include <ImplementationProxy.h>
#include "CMyHelloEcom.h"

// Maps the interface UIDs to implementation factory functions
const TImplementationProxy ImplementationTable[] =
    {
    IMPLEMENTATION_PROXY_ENTRY(0xE0009DC7, CMyHelloEcom::NewL)
    };

// Exported proxy for instantiation method resolution
EXPORT_C const TImplementationProxy* ImplementationGroupProxy
(TInt& aTableCount)
    {
    aTableCount =
        sizeof(ImplementationTable) / sizeof(TImplementationProxy);
    return ImplementationTable;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
```

Postconditions

The ECom interface has been implemented. The ECom component has a suitable resource file. Implementation factory functions have been defined in the "proxy".

See also

- File:ECom.zip: Working example code
- Using an ECOM component
- Defining a custom ECOM interface
- ECOM in Nokia Developer Wiki
- S60 Platform: ECom Plug-in Architecture
- S60 Platform: ECom Plug-in Examples

(C) Copyright Nokia 2014. All rights reserved.