



Contents

- [1 Writing a Static DLL](#)
- [2 DLL Header file.](#)
- [3 DLL Implementation file.](#)
- [4 Freezing Exports](#)

Writing a Static DLL

A static DLL is also known as shared library. They are used to share common code between different applications; only one copy is loaded into memory and shared between different applications. This places one restriction on static DLLs; they cannot have writable static data.

A static DLL is automatically loaded in the mobile RAM memory when a program that use it is started (except if it is in ROM where it is executed in place). It is also automatically unloaded when nobody needs it anymore. (Note: when a new version of a DLL is installed while it is already in use, the old version of the DLL will continue to be used *in newly launched processes as well*, until the DLL is unloaded from memory and loaded again.)

The static interface DLL provides a unique set of function within the system (i.e. two DLLs with the same exported function could not coexist in the system). Static interface DLLs have .dll file extension and are commonly used to implements application engines (i.e. UI independent code) in Symbian OS.

DLL Header file.

All methods that need to be accessed from outside the DLL need to be exported (**EXPORT_C**) in the source file (.cpp) and imported (**IMPORT_C**) by the header file (.h). Hence give IMPORT_C in front of the methods in the header file and EXPORT_C in cpp file.

```
class CMyDll : public CBase
{
public:
    IMPORT_C static CMyDll* NewL();
    IMPORT_C static CMyDll* NewLC();
    IMPORT_C void ExampleFunction();

private:
    CMyDll();
    void ConstructL();
};
```

- Hint: an EXPORT_C clause without a matching IMPORT_C may result in a *Menu: Feature not supported* message in a GUI application attempting to load the DLL upon starting up.

DLL Implementation file.

Every DLL on S60 must have an E32Dll() method as an entry point to the DLL, It is also referred to as the DLL entry point.

```
EXPORT_C TInt E32Dll( TDllReason )
{
    return KErrNone;
}

EXPORT_C void ExampleFunction( )
{
    // Do something..
}
```

Freezing Exports

To create a static DLL the **TARGETTYPE** needs to be set to dll (e.g., TARGETTYPE dll) within the project file (.mmp file).

Released versions of DLLs **should freeze their exports**, so as to ensure the backward compatibility of new releases of a library. While you are developing a DLL, you can use the **exportunfrozen** keyword in the project's mmp file, to tell the build process that exports are not yet frozen. When you are ready to freeze, remove the exportunfrozen keyword, and supply a .def file listing the exports.

To create a .def file, build the project in the normal way. A warning will be generated to the effect that the frozen .def file does not yet exist. Once the project has been built you can freeze it by calling the **freeze target** in the **makefiles**. You can do this with abld by using:

abld freeze

Either method will create the frozen .def file containing the projects exported functions. Note that the emulator targets and ARM targets have different .def file formats, so that if you build for both of these, you will need to store two def files.

Once the project is frozen, **regenerate the makefiles** so that the import library will be created directly from the **frozen .def file**.