

Comparing Symbian GUI and Console applications

This article explains the main differences between developing a full Symbian C++ GUI application and a basic console application.

Introduction

Symbian C++ GUI applications are created using the Avkon framework classes, which are themselves based on the generic Uikon layer. It is possible to create much simpler applications using Symbian's console classes - these are suitable for writing test harnesses and basic test code.

This article explains the main differences between console and full GUI applications, highlighting the frameworks that are not included by default in console programs.



Note: Prior to Symbian OS v9 (S60 3rd Edition) applications were actually DLLs, launched by an app launcher exe. From Symbian OS v9 they are exes too

GUI Applications

- A Symbian OS **GUI** application, has at least four classes: 1) An application. 2) A document. 3) An app UI. 4) An app view. Except for the view, these classes are derived from base classes in Avkon which themselves are derived from Uikon classes.
- Every control in a **GUI** application has a member called `iEikonEnv`, which is a pointer to the GUI environment, `CEikonEnv`.
- **GUI** application has control environments, so it provides file sessions (`RFs`), Active Scheduler (`CActiveScheduler`), font, graphics context (`CWindowGc`) and screen device (`CWScreenDevice`), so no need to create these thing in application.
- In a **GUI** application, you don't need to open a file server session, since the control environment already has an open `RFs` you can access with `iCoeEnv->FsSession()`.
- All processing in a **GUI** application takes place in an Active Object, but much of this, along with the creation of the Active Scheduler, is hidden by the Application Framework, hence the UI framework automatically create, install and start an Active Scheduler in **GUI** Applications.
- On S60 1st and 2nd edition apps are DLLs
 - GUI application files have `.app` filename extension and reside in a subfolder of `systemapps\`
 - The main entry point is `E32Dll()`
 - GUI programs as are launched with `apprun.exe` and have a small fixed program stack, of the order of 20k.
- On S60 3rd edition apps are exes
 - App binaries reside in `sys\bin\`. App registration and data files reside in an app-specific sub folder `\private\SID\`
 - The main entry point is `E32Main()`
 - stack size can be controlled through the MMP file.

Console applications

- Console application does not provide frameworks we take for granted in a GUI application: file sessions (`RFs`), Active Scheduler (`CActiveScheduler`), font, graphics context (`CWindowGc`) and screen device (`CWScreenDevice`). If we need these, then we must create them directly.
- In a **console** application, the **cleanup stack** must be explicitly provided, in a standard **GUI** application, it's provided by the UI framework.
- An Active Scheduler is not provided by default in **console** application, Without a valid Active Scheduler installed, any use of Active Objects will result in an `E32USER-CBase 44` panic. Hence in a **console** application or DLL, you must create and install your own Active Scheduler.

```
CActiveScheduler* scheduler = new (ELeave) CActiveScheduler();
CleanupStack::PushL(scheduler);
CActiveScheduler::Install(scheduler);
```

- In **Console** applications CONE environment (`CCoeEnv`) as a minimum is needed explicitly to add GUI controls.
- **Console** applications are not suitable for deployment onto an end users device—Locating or launching them is usually

difficult though it can be done by creating an (.rss) file.

- **Console** applications also do not have application information (.aif) files or resource (.rsc) files.
- The main entry point for **console** app (.exe) programs is E32Main()
- You can control the **stacksize** in a **console** application (.exe), through the use of the epocstacksize keyword of the .mmp.
- On S60 1st/2nd edition console applications have a .exe filename extension and reside in a **systemprograms** directory.
From S60 3rd Edition the exe is in *lsysbin*