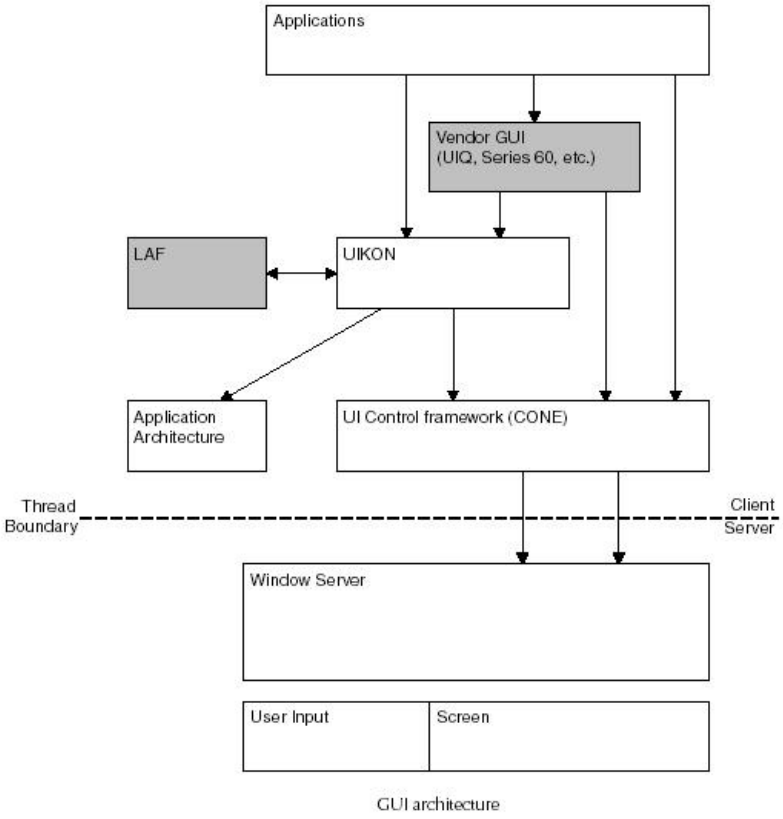
GUI Framework

GUI Framework

The main components of the GUI architecture and their relation to each other are shown in Figure below :



GUI architecture

window server

provides centralized access to the screen and user input devices across all applications. As the name implies, the window server is a server process – applications (using libraries) act as clients to this server. The window server handles the details of drawing window and control objects to the screen, as well as keeping track of which windows belong to which applications. The window server will also ensure that events such as key presses, pointer events and redraw events are routed to the correct application for handling. The window server does not enforce any particular UI policy since its commands are low level – the GUI look is handled by the upper GUI layers.

UI control framework

It is sometimes referred to as CONE (control environment). This is a library of C++ abstract classes which communicate directly with the window server via the client/server IPC channel. The UI control framework provides higher-level functionality than the window server and is more suitable for application use. The library contains no concrete controls – upper GUI layers use these base classes to derive their own specific controls. The derived classes need not worry about the details of the client/server communication with the window server since the base classes of the UI control framework handle this.

UIKON

UIKON is the core Symbian OS application framework library. While the UI control framework contains mainly abstract classes, UIKON provides a set of concrete controls and event handler classes. These classes are derived from UI control framework base

classes. UIKON also implements classes derived from the application architecture library, which will handle the basic application framework itself and non-displayrelated application behavior such as managing application documents and handling the command line.

LAF

LAF (Look and Feel) is a library that allows the appearance (e.g. size and color) of UIKON controls to be changed by a vendor without actually modifying any UIKON code. The purpose of LAF is to allow minor look and feel modifications to occur without needing to derive new controls. While having UIKON plus LAF allows customization to a certain extent, a vendor GUI layer also exists for maximum UI flexibility. This vendor layer consists of C++ classes, which derive from UIKON classes as well as directly from the UI control framework. The vendor can therefore supply its own custom controls or extend the functionality of existing UIKON controls. It can also customize application architecture-oriented behavior.

Vendor GUI layer

Applications use classes in the vendor GUI layer as well as from UIKON directly to implement the user interface.A vendor's software platform will have its own SDK with guidelines, applications should follow these guidelines when determining what classes to call. In addition to using vendor and UIKON classes, applications can create their own custom controls by deriving directly from the UI Control Framework. Also, there is nothing to prevent user programs from directly calling the window server when more screen control is desired.

Applications

An application is a polymorphic DLL (with a .app suffix) and thus cannot have static data – however, each application runs as an independent process. How is this possible? This is accomplished by invoking a process named apprun.exe and having it call the application DLL. So a separate process instance of apprun.exe exists for every open application and each application process has its own client session with the window server (through the other GUI layers).