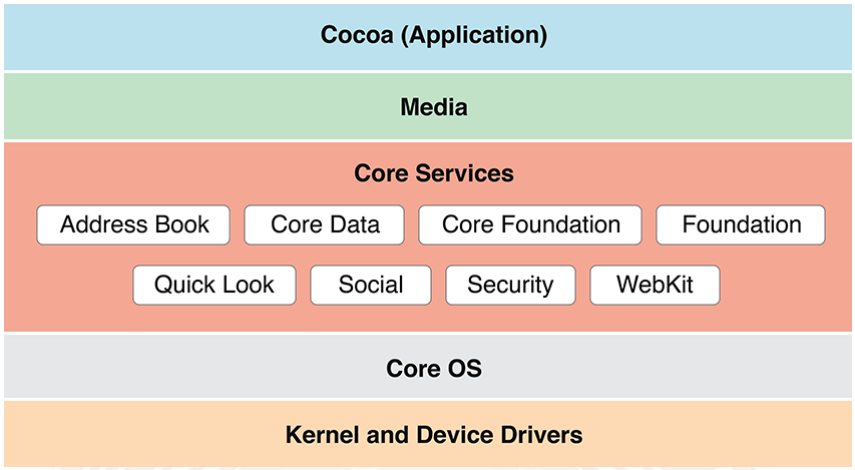
1. THE IOS CORE SERVICES LAYER

The iOS Core Services layer provides much of the foundation on which the previously referenced layers are built and consists of the following frameworks.



Address Book framework (AddressBook.framework)

The Address Book framework provides programmatic access to the iPhone Address Book contact database allowing applications to retrieve and modify contact entries.

CFNetwork Framework (CFNetwork.framework)

The CFNetwork framework provides a C-based interface to the TCP/IP networking protocol stack and low level access to BSD sockets. This enables application code to be written that works with HTTP, FTP and Domain Name servers and to establish secure and encrypted connections using Secure Sockets Layer (SSL) or Transport Layer Security (TLS).

Core Data Framework (CoreData.framework)

This framework is provided to ease the creation of data modeling and storage in Model-View-Controller (MVC) based applications. Use of the Core Data framework significantly reduces the amount of code that needs to be written to perform common tasks when working with structured data in an application.

Core Foundation Framework (CoreFoundation.framework)

The Core Foundation is a C-based Framework that provides basic functionality such as data types, string manipulation, raw block data management, URL manipulation, threads and run loops, date and times, basic XML manipulation and port and socket communication.

Additional XML capabilities beyond those included with this framework are provided via the libXML2 library. Though this is a C-based interface, most of the capabilities of the Core

Foundation framework are also available with Objective-C wrappers via the Foundation Framework.

Core Media Framework (CoreMedia.framework)

The Core Media framework is the lower level foundation upon which the AV Foundation layer is built. Whilst most audio and video tasks can, and indeed should, be performed using the higher level AV Foundation framework, access is also provided for situations where lower level control is required by the iOS application developer.

Core Telephony Framework (CoreTelephony.framework)

The iOS Core Telephony framework is provided to allow applications to interrogate the device for information about the current cell phone service provider and to receive notification of telephony related events.

EventKit Framework (EventKit.framework)

An API designed to provide applications with access to the calendar and alarms on the device.

Foundation Framework (Foundation.framework)

The Foundation framework is the standard Objective-C framework that will be familiar to those that have programmed in Objective-C on other platforms (most likely Mac OS X). Essentially, this consists of Objective-C wrappers around much of the C-based Core Foundation Framework.

Core Location Framework (CoreLocation.framework)

The Core Location framework allows you to obtain the current geographical location of the device (latitude and longitude) and compass readings from with your own applications. The method used by the device to provide coordinates will depend on the data available at the time the information is requested and the hardware support provided by the particular iPhone model on which the app is running (GPS and compass are only featured on recent models). This will either be based on GPS readings, Wi-Fi network data or cell tower triangulation (or some combination of the three).

Mobile Core Services Framework (MobileCoreServices.framework)

The iOS Mobile Core Services framework provides the foundation for Apple's Uniform Type Identifiers (UTI) mechanism, a system for specifying and identifying data types.

A vast range of predefined identifiers have been defined by Apple including such diverse data types as text, RTF, HTML, JavaScript, PowerPoint .ppt files, PhotoShop images and MP3 files.

Store Kit Framework (StoreKit.framework)

The purpose of the Store Kit framework is to facilitate commerce transactions between your application and the Apple App Store. Prior to version 3.0 of iOS, it was only possible to charge a customer for an app at the point that they purchased it from the App Store. iOS 3.0 introduced the concept of the "in app purchase" whereby the user can be given the option make additional payments from within the application. This might, for example, involve implementing a subscription model for an application, purchasing additional functionality or even buying a faster car for you to drive in a racing game.

SQLite library

Allows for a lightweight, SQL based database to be created and manipulated from within your iPhone application.

System Configuration Framework (SystemConfiguration.framework)

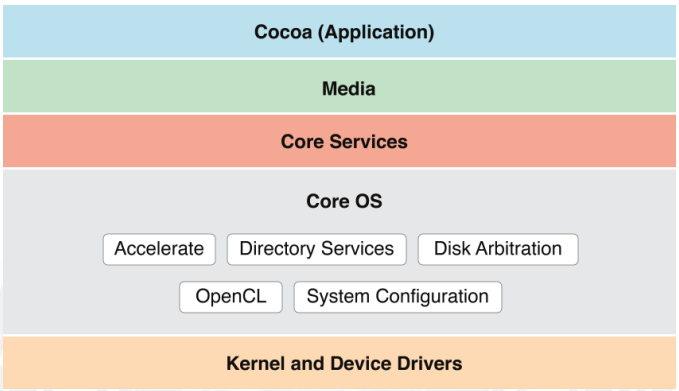
The System Configuration framework allows applications to access the network configuration settings of the device to establish information about the "reachability" of the device (for example whether Wi-Fi or cell connectivity is active and whether and how traffic can be routed to a server).

Quick Look Framework (QuickLook.framework)

One of the many new additions included in iOS 4, the Quick Look framework provides a useful mechanism for displaying previews of the contents of files types loaded onto the device (typically via an internet or network connection) for which the application does not already provide support. File format types supported by this framework include iWork, Microsoft Office document, Rich Text Format, Adobe PDF, Image files, public.text files and comma separated (CSV).

2. THE IOS CORE OS LAYER

The Core OS Layer occupies the bottom position of the iOS stack and, as such, sits directly on top of the device hardware. The layer provides a variety of services including low level networking, access to external accessories and the usual fundamental operating system services such as memory management, file system handling and threads.



Accelerate Framework (Accelerate.framework)

Introduced with iOS 4, the Accelerate Framework provides a hardware optimized C-based API for performing complex and large number math, vector, digital signal processing (DSP) and image processing tasks and calculations.

External Accessory framework (ExternalAccessory.framework)

Provides the ability to interrogate and communicate with external accessories connected physically to the iPhone via the 30-pin dock connector or wirelessly via Bluetooth.

Security Framework (Security.framework)

The iOS Security framework provides all the security interfaces you would expect to find on a device that can connect to external networks including certificates, public and private keys, trust policies, keychains, encryption, digests and Hash-based Message Authentication Code (HMAC).

System (LibSystem)

As we have previously mentioned, the iOS is built upon a UNIX-like foundation. The System component of the Core OS Layer provides much the same functionality as any other UNIX like operating system. This layer includes the operating system kernel (based on the Mach kernel developed by Carnegie Mellon University) and device drivers.

The kernel is the foundation on which the entire iOS is built and provides the low level interface to the underlying hardware. Amongst other things the kernel is responsible for memory allocation, process lifecycle management, input/output, inter-process

communication, thread management, low level networking, file system access and thread management.

As an app developer your access to the System interfaces is restricted for security and stability reasons. Those interfaces that are available to you are contained in a C-based library called LibSystem. As with all other layers of the iOS stack, these interfaces should be used only when you are absolutely certain there is no way to achieve the same objective using a framework located in a higher iOS layer.