Carbon Overview

Carbon

2005-11-09

Contents

Introduction to Carbon Overview

Originally designed to provide a gentle migration path for developers transitioning from Mac OS 9, Carbon is a collection of C programming interfaces that let you implement basic application functionality such as the user interface, event handling, file management, and so on.

Note: A programming interface is the set of functions and data structures defined by one piece of software, such as an operating system service, for use by client software, such as applications and device drivers. For example, you would access one programming interface to enable your application to print and another to manipulate your application's menus

This document describes Carbon's place in Mac OS X and gives overviews of the Carbon interfaces. It also describes a wide variety of other programming interfaces that Carbon applications can use, supporting everything from video playback to alternate text input.

As Carbon is a C interface, you can also use all the standard C library APIs; however, Mac OS X often has superior replacements (for example, Unicode string manipulation APIs versus the ASCII-related APIs, such as `strcmp` in the standard C library).

Who Should Read This Document?

You should read this document if you are new to Mac OS X and would like to write Mac OS X applications using procedural C or C++.

Organization of This Document

This document contains two chapters:

- "Carbon Basics" (page 7) briefly describes how Carbon fits into Mac OS X and the tools available to build Carbon applications.
- "The Carbon Factory Tour" (page 9) gives an overview of Carbon managers and services, including nonCarbon specialty services.
- "Legacy Interfaces" (page 19) describes managers and services that have been superseded by newer technologies. If you are a new developer, this chapter is of historical interest only.

See Also

When you are ready to explore Carbon programming, see *Getting Started with Carbon* to determine which documents to start reading.

Carbon Basics

This chapter describes the basics of Carbon, how it fits into Mac OS X, and the tools you use to build Carbon applications.

What's in Carbon?

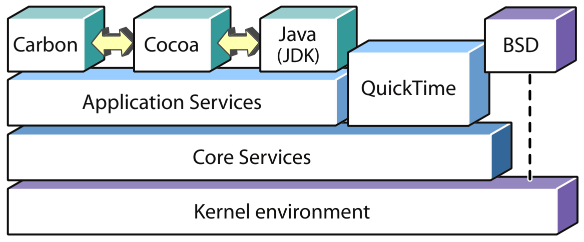
Carbon contains thousands of functions, data structures, and constants. Related functions and data structures are organized into functional groups, usually referred to as managers or services. For example, the Window Manager contains functions and data structures that let you create, remove, and otherwise manipulate application windows.

As a general rule, Carbon interfaces are those that are part of the Carbon umbrella framework; that is, those that are included when you include the `Carbon.h` header.

This document also describes a number of managers or services that you can call from a Carbon application (for example, Quartz and QuickTime). Some of these technologies use Objective-C, which may require you to write some wrapper code.

Carbon and Mac OS X

As shown in the following figure, Carbon is one of several application environments available on Mac OS X.



These other environments include:

- Cocoa. The object-oriented interface for writing Mac OS X applications in Objective-C.
- Java. A JDK-compliant virtual machine for running Java applications.

These environments depend on the same application and core services for their operation, and the underlying services rely on Darwin (Apple's open-source core operating system) and the Mach kernel.

Each environment (including the optional BSD command-line environment) has advantages and disadvantages, but for C and C++ programmers, Carbon is the best choice.

Building Carbon Applications

Xcode and Interface Builder are Apple's development tools for building Mach-O-based applications on Mac OS X. Both tools support Carbon, and they come free with Mac OS X, making them excellent choices for new developers targeting Mac OS X. Also Xcode is currently the only development environment that allows compiling for both PowerPC and Intel-based Macintosh computers.

Note: Mach-O is the native executable format on Mac OS X. While Carbon supports the PEF or CFM-based executable formats, you should not use this format unless you need to be compatible with pre-Mac OS X systems (not likely).

Xcode and Interface Builder work in conjunction with each other to make building applications easier than ever:

- Xcode. The main development environment; it lets you create and assemble the components of your application. When you create a new Carbon application, Xcode uses a Carbon Event-based template that automatically provides basic event-handling support. Xcode also makes it easy to build universal binaries, which support both PowerPC and Intel-based Macintosh hardware.
- Interface Builder. A WYSIWYG tool that lets you lay out user interfaces in a simple, intuitive manner. These interface templates are stored as a .nib file, which your application can then access to create its windows, menus, and other elements.

Xcode uses open-source developer tools (such as the GCC compiler) under the hood. If you are comfortable doing so, you can run these tools yourself from the command line using the Terminal application, writing Make files and so on.

The Carbon Factory Tour

Carbon contains many programming interfaces. What interfaces you will require depends on the type of applications you want to write, but as is often the case, some are more commonly used than others. This section divides up the Carbon interfaces according to usage (from fundamental to esoteric) and gives some useful information about each one. To learn more about a particular interface, you can use the Documentation Help feature integrated into Xcode. You can also consult the technical documentation on the Apple developer web site.

Carbon interfaces are defined as those accessed through the Carbon umbrella framework. See the header `Carbon.h` to see which interfaces are included.

Note: Carbon also includes a number of legacy interfaces that you should not use unless you are supporting older systems. "Legacy Interfaces" (page 19) lists these older interfaces.

The Starter Kit

This section covers the interfaces most likely to be called by a Macintosh application. These managers and services provide the basic user interface as well as fundamental features, such as the ability to save and print files. You can build basic but fully functional Carbon applications using the interfaces described here.

Note: If Carbon were an interface to build automobiles, the Starter Kit interfaces would define the engine, wheels, brakes and other essential components, as well as the user interface (dashboard, steering wheel, gas and brake pedals, and so on).

The Toolbox Interfaces

The following interfaces (sometimes called the Human Interface Toolbox or HIToolbox) are grouped together because they generally work together to create and manage the user interface. In most cases, Interface Builder and the Interface Builder Services programming interface can handle the creation and control of the basic user interface. However, to accomplish more esoteric tasks, you may need to call additional functions in the Human Interface Toolbox interfaces:

- **HIView/Control Manager**. Lets you create and manipulate views, which are building blocks for user interface elements. All standard controls (buttons, scroll bars, sliders, and so on) are views, as is content displayed in a menu. Special views also exist for displaying text, HTML, and QuickTime content. HIView is superseding the older Control Manager, which let you manipulate controls. At the implementation level, however, HIViews and controls are identical objects in memory and can be used interchangeably.
- **Dialog Manager**. Lets you create and manipulate dialogs, which are windows that either prompt you for input or display information. Interface Builder and Carbon event handlers mostly supersede the Dialog Manager, except in the cases where you want to put up a standard alert dialog box or sheet.

- **Menu Manager.** Lets you create and manipulate menus.
- **Window Manager.** Lets you create and manipulate windows, which are the user's primary means of interacting with your software.
- **HIToolbar.** Lets you create and manipulate toolbars.
- **HITheme/Appearance Manager.** Provides the underlying support for themes, which unify the appearance of human interface objects in your application, including alert icons, controls, background colors, dialogs, menus, and windows. When you draw nonstandard user interface elements with HITheme, their appearance will continue to match the standard elements even if the theme changes (for example, from Aqua to Graphite). The Appearance Manager is the older QuickDraw-centric version of HITheme.
- **Carbon Event Manager.** Controls the event model for Macintosh applications. Note that Carbon also contains an older event-handling system (simply called the Event Manager), which is included only to assist legacy applications moving to Mac OS X.
- **Interface Builder Services.** Lets your code access the user interface elements created with Interface Builder and stored in a nib file. While Interface Builder is not required for creating your user interface, it simplifies the process and makes localization easier as well.

The User Interface

While the HIToolbox interfaces are flexible enough to let you do just about anything with your user interface, that doesn't mean you should. When designing and laying out your user interface, you need to follow the *Apple Human Interface Guidelines*. Just as adherence to common rules and customs when designing steering wheels and dashboards makes driving more pleasant and less confusing, following the Apple guidelines lets your application provide the best possible experience for your users.

If you use Interface Builder, adhering to the Apple Guidelines is that much easier; you can have it automatically tell you when various items in a window are spaced correctly.

Accessibility

While the standard user interface is sufficient for most users, those who have disabilities may need alternate methods to interact with your application. Apple provides a number of special accessibility features to meet this need. If you plan to sell your application to governmental agencies in the United States, your application must support accessibility.

Accessibility is not just a benefit for disabled users. An application that supports accessibility is also easier to test, and it makes it easier to provide explanatory help for all users.

See *Accessibility Overview* to learn more about accessibility on Mac OS X, and *Accessibility Programming Guidelines for Carbon* to learn what you need to do to support accessibility in Carbon applications.

Under the Hood

The following interfaces work behind the scenes, as it were, to provide the basic functionality that you expect from most applications:

- **Carbon Printing Manager.** Lets you print from Macintosh applications.

- **Navigation Services.** Creates a standard user interface for opening and saving documents.
- **File Manager.** Lets you read and write data to storage media (hard drives, USB drives, and so on).
- **HIObject**. A "base class" that provides the underlying support for HIViews. You probably won't need to access HIObject APIs directly unless you are creating custom HIViews.
- **HIArchive.** Provides an easy way to store and retrieve data and HIObjects (including HIViews). HIArchives are stored in a flattened format that you can save on disk or pass to other applications or services as desired.
- **Bundle Services.** Lets you access data stored in a bundle file hierarchy, which is the standard method of packaging applications in Mac OS X. Bundle Services is part of Core Foundation.
- **String Services.** Allows basic manipulation of Unicode strings. String Services is part of Core Foundation.
- **Multilingual Text Engine (MLTE).** Provides basic text display and formatting features. Because it is Unicode-based, MLTE can easily handle other languages and script systems. MLTE also provides support for spellchecking and accessing the standard system font panel.
- **Quartz.** Handles all the 2D drawing to the Macintosh screen. Quartz is a powerful imaging technology that provides easy access to features such as transparency, layers, path-based drawing, offscreen rendering, advanced color management, anti-aliased rendering, and PDF document creation, display, and parsing.
- **HIShape.** A Quartz-compatible interface for creating and manipulating arbitrary graphical shapes.
- **HIGeometry.** A Quartz-compatible interface for defining and manipulating basic graphical forms (points and rectangles).
- **Uniform Type Identifers.** Not a technology, but a standard for identifying data and file types that eliminates having to keep track of all the possible ways to tag a particular type (for example, using file extensions, MIME types, OSTypes, and so on). Navigation Services and the Pasteboard Manager are two technologies that use uniform type identifiers.

The Expansion Pack

This section lists interfaces that are desirable, but not necessary for most applications. Full-featured commercial applications usually adopt a number of these interfaces.

Note: Continuing the automobile metaphor, these interfaces would add useful features that may be essential in some cases, such as power-steering, windshield wipers, and air conditioning.

- **Pasteboard Manager.** Lets the user copy items to and from the Clipboard for cut-and-paste operations. More generally, you can use pasteboards for any sort of data interchange. For example, the Drag Manager uses pasteboards to transfer drag and drop data.
- **Drag Manager.** Used for implementing drag and drop between applications. For example, the user can select text in one application and then, rather than copying and pasting, simply drag the text into the window of another application.
- **Icon Services.** Provides a simplified way to present icons in your application. Instead of storing every type of icon you need with your application, you can obtain commonly used icons through Icon Services. Doing so minimizes the amount of work you have to do and increases system efficiency.
- **Folder Manager.** Lets you find, create, and otherwise manipulate specific folders such as a user's home directory or the system's Fonts folder.

- **Alias Manager.** Lets you create and resolve aliases to files and folders. An alias is a link or reference to an application, folder, or file.
- **Open Scripting Architecture.** Describes the interfaces to control or automate the actions of one or more applications. You use the **Apple Event Manager** portion to interpret and react to events received from other applications, and **AppleScript** is the scripting language that you use to describe what actions to take. For example, if your application might be used in an automated workflow, where multiple applications manipulate a file in turn, you should make your application Apple Event savvy so it can act upon commands sent to it from an external script.

  Note: The Apple Event Manager is different from the Carbon Event Manager. While there is some overlap in their capabilities, Carbon events are typically received from the user interface or operating system, while Apple events come from scripts or other applications.

- **Core Foundation.** A collection of object-based interfaces that can handle data types and various system services. Bundle Services and String Services (already described in "The Starter Kit") are Core Foundation services. Other Core Foundation interfaces handle plug-ins, preferences, and so on. For the complete list of Core Foundation interfaces see *Core Foundation Framework Reference*. To learn how to use Core Foundation APIs, see *Getting Started with Core Foundation*.
- **Search Kit.** Provides powerful text searching capability for your application. Apple's Spotlight technology is built on top of Search Kit to provide content searching in Finder, Mail, and the Spotlight menu.
- **Gestalt Manager.** Lets your application determine specific information about the system, its interfaces, or the underlying hardware. For example, you can call the Gestalt Manager to determine what version of a particular technology is installed. Doing so lets you avoid calling functions that may not be available on a particular computer.
- **Notification Manager.** This interface allows applications in the background to notify the user (for example, after finishing a lengthy calculation). A primary use of the Notification Manager is to notify the Dock about changes to the application.
- **Process Manager**. Lets you obtain useful information about running processes.
- **Multiprocessing Services.** Lets you create preemptively scheduled execution tasks (threads) in your application. If your application might want to perform several actions simultaneously (such as downloading files or performing background calculations), you should consider adopting Multiprocessing Services. Tasks created with this interface will automatically take advantage of multiple processors, if present.
- **Thread Manager.** Lets you create cooperatively scheduled threads in your application. This interface is generally not as useful as Multiprocessing Services. However, if you need a certain amount of control over when your threads execute, you should consider the Thread Manager.
- **Quartz Services.** Lets you access low-level features of the Mac OS X Window Server. For example, you can use Quartz Services to change display modes, monitor configurations, and so on.

Specialty and NonCarbon Interfaces

These are more esoteric interfaces that you generally would not use unless you were interested in creating specific types of applications. Some provide specialized features, while others expand on basic functionality (such as text manipulation). You use these interfaces to create highly sophisticated applications that take full advantage of the system software.

Many of these interfaces are not part of the Carbon framework, but can be called from Carbon applications.

Some of these interfaces have Objective-C APIs, which will require you to write some Objective-C code. For details on how to call Objective C code from C code, see *Carbon-Cocoa Integration Guide*.

Note: For automobiles, these features would be for specific types of cars: turbocharging and tight suspensions for sports cars, flatbeds and towhooks for trucks, leather seats and soft suspensions for luxury cars, and so on.

QuickTime

QuickTime is Apple's multimedia programming interface. You use QuickTime to create and play file-based or streaming movies, virtual reality environments, sounds, and music files. The QuickTime programming interface is broad and very powerful. To gain a better understanding of its features, read *Getting Started with QuickTime* and *QuickTime Overview*.

Core Video

Core Video complements QuickTime as it provides a way to access individual frames in the video pipeline. You can then manipulate the frames using OpenGL or Core Image. For example, if you want to add filter effects to the video, or map the video onto a shape, you can use Core Video to do so. See *Core Video Programming Guide* for more information.

Core Audio

Core Audio is the audio architecture for Mac OS X. It provides a broad range of audio and MIDI services for hardware and application developers, such as state-of-the-art audio recording and playback, digital signal processing, MIDI sequencing, low-level access to audio devices, software-based sound synthesis, and much more.

Color, Images, and Print Production

These interfaces are for applications that create and manipulate images, such as a photo retouching program:

- **Color Picker Manager** lets you bring up a simple user interface for choosing colors, which can be useful for paint programs as well as any application that allows the user to customize colors.
- **ColorSync** is an Apple technology that ensures consistent color calibration across different applications and hardware. For example, when using ColorSync, users can be sure that the particular shade of green they see on their monitor is as close as possible to what they will get when the local print house prints their brochure.
- **Picture Utilities** are used to obtain information about a graphic image, such as the colors used, its resolution, and any comments that may be included.
- **Core Image** provides access to built-in image filters for both still images and video and provides support for custom filters and near real-time processing. Core Image is an Objective-C API.

3D Graphics

For high-quality 3D graphics, the interface of choice is **OpenGL**. In actuality this is an industry-standard interface and not part of Carbon, but it is fully compatible with Carbon (just make sure you link to the OpenGL framework when you build). You can use OpenGL's 3D rendering capabilities for everything from medical imaging to virtual reality to incredibly photorealistic games. See *OpenGL Programming Guide for Mac OS X* for more information about how to use OpenGL in Mac OS X.

HTML

To render HTML in Carbon windows, you can use **Web Kit.** HIView has a special web view that can display HTML acquired with Web Kit. See *WebKit C Reference* for more details.

Disc Recording and Playback

These interfaces let your application interact with CDs or DVDs.

- To burn data or audio to a CD or DVD from your application, use the **Disc Recording** API.
- To access and play DVD video content, use the **DVD Playback** API.

Sync Services

If your application needs to synchronize some of its data with a device, such as an iPod or PDA, you should use **Sync Services** to do so. You can also synchronize separate Macintosh computers if the user has a .Mac account. Sync Services is an Objective-C API.

Speech

These interfaces let your application speak text or recognize speech:

- **Speech Recognition Manager** lets your application recognize spoken commands. For example, you can navigate between windows, open files, or run scripts solely through voice commands. It can be especially useful to activate commands that would normally require navigating deeply nested menus or multiple dialogs.
- **Speech Synthesis Manager** lets your application speak lines of text using a number of different voices. Note that the speech recognition and speech synthesis interfaces use the same English dictionary, which allows them to work in conjunction with each other. For example, you could use the Speech Synthesis Manager to determine how to pronounce a word, so the Speech Recognition Manager can recognize it more easily.

Text and International Services

Most of these interfaces are only for developers writing text-intensive applications. For basic text input and display, the Multilingual Text Engine provides a much simpler interface for most of the same functionality.

- **ATSUI** is the interface for drawing Unicode text. It allows precise control over all aspects of the text, from kerning to ligatures to bidirectionality.
- **ATS Types** interface defines data types and callback functions used by ATSUI and other text interfaces.
- **Date, Time, and Measurement Utilities** contain functions to obtain and manipulate date, time, location, and other values that may need to be localized for different countries or regions.
- **Dictionary Manager** provides an easy interface to access dictionary files. For example, if your application contained a spell checker, it could use this interface to look up words in a dictionary file. Similarly, text input methods that require looking up words in a file could also use this interface.
- **Font Manager.** Lets you manipulate the fonts that your application uses to display or print text.
- **International Resources** contain structures and constants that are used for localizing text to different countries or regions. In most cases, you won't need to access this interface yourself, because other text interfaces will access it for you.
- **Text Services Manager** provides support for text input methods. For example, editing some types of non-Western text requires close cooperation between the application and the Text Services Manager to allow text input methods to access the application's windows. If you use the Multilingual Text Engine, all Text Services Manager support is handled automatically.
- **FontSync** allows you to synchronize fonts available on different computers or printers to prevent font mismatches. For example, two fonts on different computers may have the same name but not be identical. FontSync can attempt to match fonts based on content rather than name, thus minimizing the possibility of a mismatch when a text file is moved from one computer to another.
- **Language Analysis Manager** allows your application to manage language analysis engines (stored as plug-ins). These engines are typically used with text input methods to isolate meaningful words or characters. For example, for Japanese text input you may use a language analysis engine to interpret the keystrokes the user enters, so you can display the Kanji characters that match their meaning.
- **Text Encoding Conversion Manager** allows you to change text from one encoding to another. This conversion can be useful for text going to or from the Internet, where many different text encodings exist. For example, to read text streamed over a network from a Windows computer, you may need to convert it from the Windows text encoding to the corresponding one for Macintosh computers. Similarly, if you are handling input methods or file systems that only support the older Mac OS encodings, you can use the Text Encoding Converter to convert between them and Unicode.
- **Unicode Utilities** let you perform basic manipulation of Unicode strings. Note that Core Foundation String Services provide similar functionality and are more portable across Mac OS X execution environments.
- **Ink Services** is an interface that lets users enter text using a stylus on a graphics tablet. The text is automatically recognized and translated into keystrokes that your application can then interpret.

Internet and Networking

If your application uses or enables network access, you may need to use one of the following managers or services:

- Core Foundation's **CFNetwork Services** provides interfaces for basic networking tasks, such as working with BSD sockets, managing information about remote hosts, and working with HTTP, HTTPS, and FTP servers. See *CFNetwork Programming Guide* for more information.
- **Network Services Location Manager**, which provides an easy way to find network services on a local network.
- **Web Services**, which provides an interface for transferring data over the internet using standard protocols such as HTTP and XML.
- **Internet Config**, which is used to access Internet networking preferences from a global repository on a user's machine. For example, Internet Config stores the user's default browser selection, so if another application needs to launch a browser (when a URL is clicked), it can easily determine which one to activate.

Mac OS X also supports standard networking security features, such as SSL/TLS. See "Security" (page 16) for more information.

Security

Mac OS X supports a number of standard security services (certificates, an authentication interface, SSL/TLS for networking, and so on) as well as Apple-specific features, such as keychains, which allow you store and access multiple passwords using a master password. For an overview of basic security concepts and the security services available in Mac OS X, see *Security Overview*. *Getting Started with Security* describes documentation available for implementing security features.

Low-Level Tweaking

Mac OS X is designed to shield applications from low-level workings of the system. However, if you are writing driver-level code that needs to talk directly to hardware (such as a video card), you need to use **I/O Kit.** Most applications don't need this level of control and should not be at all dependent on hardware.To that end, you should use I/O Kit only if you are sure you need it.

Utility Interfaces

This section covers utility interfaces that may be useful, depending on the application. These managers and services aren't particularly related to any technology or functionality:

Note: If Carbon were for cars, these interfaces would add possibly useful features, such as diagnostic self-checking (warning lights), headlight wipers, cup holders, curb feelers, and GPS units.

- **Finder.** This interface contains a number of structures that are useful for giving information to the Finder.
- **Debugger Services.** Contains functions that can assist you in debugging your application. In most cases, however, you should begin with the debugging facilities available with Xcode before using this interface.

- **Assert Macros.** Contains debugging macros that you might find useful. Many Apple code samples use these macros, so it's useful to understand how they work. These macros are defined in `AssertMacros.h`, which you can find in the `/usr/include` directory (use Terminal to access).
- **Mathematical and Logical Utilities.** Contains functions and constants for mathematical and logical operations (for example, `FloatToFixed`, `pi`, and `BitAnd`).

Legacy Interfaces

This chapter covers Carbon interfaces that are mostly of historical interest. These interfaces were included to assist developers porting legacy Mac OS code to Mac OS X. For new developers, these interfaces may be useful only to gain perspective on the evolution of Macintosh system software. In most cases, the functionality of these managers is covered in a newer interface

- **Code Fragment Manager.** Written to allow the system and applications to prepare and execute Preferred Executable Format (PEF) binaries, the native executable file format for Classic Mac OS PowerPC applications. While Mac OS X supports PEF binaries, the native executable file format is called Mach-O.
- **Collection Manager.** This interface let you create abstract data types to store related information together. You should use the Core Foundation interface instead.
- **Component Manager.** Older Mac OS programs often loaded special format plug-ins using the Component Manager. On Mac OS X, you should use the Plug-In Services interface in Core Foundation.
- **Device Manager.** Older Mac OS programs accessed hardware through the Device Manager. On Mac OS X, you should use the I/O Kit instead.
- **Display Manager.** Provided a way to manipulate multiple monitor locations and resolutions from your application. Deprecated in Mac OS X v10.4. Now replaced by the CGDirectDisplay API in Quartz Services..
- **Event Manager.** Originally written to handle the older cooperative multitasking event model. The Carbon Event Manager is more powerful and is a core component of the Human Interface Toolbox.
- **HTML Rendering Library.** Essentially, this interface allowed you to render text and images in a window as if it were in a browser. It provided support for such design elements as border and scroll bars, as well as for navigation using URL links. Deprecated In Mac OS X v10.4. Replaced by WebKit.
- **Keychain Manager.** Now replaced by Keychain Services. See *Security Overview*for more information.
- **List Manager.** Formerly used to create columned lists of data (similar to the list view option in a Finder window), you should now use the data browser control (part of the Control Manager).
- **Low Memory Accessors.** These functions allowed you to access useful system information (such as the location of the mouse) stored as global variables in so-called "low memory." This practice of accessing low memory directly was questionable even then, and certainly not suggested now. In almost all cases, better, safer functions exist in other interfaces for obtaining the same information.
- **Memory Management Utilities** are used for specialized low-level memory operations, such as direct memory access (DMA) and those that take place at the interrupt level. Many of the functions in this interface aren't even supported in Carbon, and it is unlikely that you will need to use the ones that are. Mostly deprecated in Mac OS X v10.4.
- **Memory Manager.** This interface allowed pre-Mac OS X applications to allocate and manage application memory to avoid fragmentation or low-memory conditions. Mac OS X handles any memory management for you transparently, so you only need to use `malloc` or `calloc` and `free` to allocate and free memory respectively.
- **Mixed Mode Manager.** Originally designed to allow PowerPC code to call emulated 68K code (and vice versa), the Mixed Mode Manager is now used to allow CFM-based PEF code to call Mac OS X's native Mach-O code and vice versa. If you build Mach-O-based, Mac OS X applications, you don't need this interface.

- **Open Transport** was Carbon's low-level networking interface. Typically you need this interface only if you are implementing network protocols (such as TCP/IP) in your application or driver. Deprecated in Mac OS X v10.4. Use CFNetwork or socket library APIs instead.
- **Palette Manager.** This interface allowed to you specify the contents of palettes, which let you specify the colors to use to render an image. For example, if a picture contained mostly dark colors, grays and blues, you could index the palette to use only those colors, eliminating those that didn't appear (yellows, bright red). Using a fixed selection of colors allowed you to reduce the size of the image, because not as much information was needed to render it. The Palette Manager was useful in the days when, due to memory constraints, many computers could display only a limited number of colors.
- **Pascal String Utilities.** Contains functions for manipulating Pascal strings. In most cases you will be using Unicode Strings on Mac OS X, but if you need to manipulate Pascal strings, this is where to go. Deprecated in Mac OS X v10.4 and later.
- **Power Manager** allows you to control power management features. For example, a screen saver program could use this interface to put the PowerBook to sleep after a given time. Other applications may want to monitor the battery status of the PowerBook. Mostly deprecated in Mac OS X v10.4. Use I/O Kit instead.
- **QuickDraw.** The original 2D graphics drawing interface. Deprecated in Mac OS X v10.4 and later. Superseded by Quartz and Core Graphics.
- **QuickDraw Text.** Superseded by MLTE or, if you need more complex text manipulations, Apple Type Services for Unicode Imaging (ATSUI).
- **The Resource Manager.** Let you access information in a bundle hierarchy stored as an old-style Mac OS resource (that is, in a data fork-based file with the .rsrc extension). The Resource Manager complements Bundle Services in allowing access to images, text, and other resources stored in a bundle.
- **Scrap Manager.** The original interface to handle cut-and-paste operations, this technology has been superseded by the Pasteboard Manager.
- **Script Manager** allows you to handle older Mac OS script systems (or text encodings), providing support for text input and display processing. On Mac OS X you will usually be handling Unicode text, so you probably won't need this interface.
- **SCSI Manager.** Originally written to communicate with SCSI devices. On Mac OS X you should use the I/O Kit instead.
- **TextEdit.** An interface to perform basic text manipulation. Its replacement, the Multilingual Text Engine, offers significant additional features, such as Unicode support.
- **Text Utilities** let you perform basic manipulation of non-Unicode text strings. If Mac OS X is your primary platform, you will likely be using Unicode strings, which means you should investigate the Unicode Utilities or Core Foundation String Services instead.
- **Time Manager.** Lets you set timers to perform periodic actions or to have actions occur at a particular time. Although timers in the Carbon Event Manager can provide some of the same functionality, the Time Manager is useful for cases where you need something to happen outside of your usual event loops. Mostly deprecated in Mac OS X v10.4 and later.
- **URL Access Manager**, which provides an easy interface for uploading and downloading files to and from a given URL. Deprecated in Mac OS X v10.4 and later.

Document Revision History

This table describes the changes to *Carbon Overview*.

Date	Notes
2005-11-09	Updated to reflect Carbon technologies as of Mac OS X v10.4.