EXHIBIT
1070
R. Surati
12/01/2022

The Wayback Machine - https://web.archive.org/web/20100525020133/http://developer.apple.com:80/mac/libr
Mac OS X Reference Library Apple Developer
Search
Search Mac OS X Reference Library

Download IconCompanion File PDF

- Table of Contents

Next

Introduction

Bundles are a fundamental technology in Mac OS X and iPhone OS that are used to encapsulate code and resources. Bundles simplify the developer experience by providing known locations for needed resources while alleviating the need to create compound binary files. Instead, bundles use directories and files to provide a more natural type of organization—one that can also be modified easily both during development and after deployment.

To support bundles, both Cocoa and Core Foundation provide programming interfaces for accessing the contents of bundles. Because bundles use an organized structure, it is important that all developers understand the fundamental organizing principles of bundles. This document provides you with the foundation for understanding how bundles work and for how you use them during development to access your resource files.

Organization of This Document

This document contains the following chapters:

- "About Bundles" introduces the concept of bundles and packages and how they are used by the system.
- "Bundle Structures" describes the structure and contents of the standard bundle types.
- "Accessing a Bundle's Contents" shows you how to use the Cocoa and Core Foundation interfaces to get information about a bundle and its contents.
- "Document Packages" describes the notion of document packages (which are loosely related to bundles) and how you use them.

See Also

Although the information in this document applies to all types of bundles, if you are working with more specialized types of bundles (such as frameworks and plug-ins), you should also consult the following documents:

- *Framework Programming Guide* provides detailed information about creating and using custom frameworks.
- *Code Loading Programming Topics for Cocoa* provides information about writing plug-ins using the Objective-C language.
- *Plug-ins* provides information about writing plug-ins using the C language.

Next

Last updated: 2009-07-14

DOCKET ALARM
Find authenticated court documents without watermarks at docketalarm.com.

Did this document help you? Yes It's good, but... Not helpful...

Shop the Apple Online Store (1-800-MY-APPLE), visit an Apple Retail Store, or find a reseller. Mailing Lists | RSS Feeds

Copyright © 2010 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

The Wayback Machine - https://web.archive.org/web/20100518063556/http://developer.apple.com:80/mac/li…

About Bundles

Bundles are a convenient way to deliver software in Mac OS X and iPhone OS. Bundles provide a simplified interface for end users and at the same time provide support for development. This chapter provides an introduction to bundles and discusses the role they play in Mac OS X and iPhone OS.

Bundles and Packages

Although bundles and packages are sometimes referred to interchangeably, they actually represent very distinct concepts:

- A **package** is any directory that the Finder presents to the user as if it were a single file.
- A **bundle** is a directory with a standardized hierarchical structure that holds executable code and the resources used by that code.

Packages provide one of the fundamental abstractions that makes Mac OS X easy to use. If you look at an application or plug-in on your computer, what you are actually looking at is a directory. Inside the package directory are the code and resource files needed to make the application or plug-in run. When you interact with the package directory, however, the Finder treats it like a single file. This behavior prevents casual users from making changes that might adversely affect the contents of the package. For example, it prevents users from rearranging or deleting resources or code modules that might prevent an application from running correctly.

Note: Even though packages are treated as opaque files by default, it is still possible for users to view and modify their contents. On the contextual menu for package directories is a Show Package Contents command. Selecting this command displays a new Finder window set to the top level of the package directory. The user can use this window to navigate the package's directory structure and make changes as if it were a regular directory hierarchy.

Whereas packages are there to improve the user experience, bundles are geared more toward helping developers package their code and to helping the operating system access that code. Bundles define the basic structure for organizing the code and resources associated with your software. The presence of this structure also helps facilitate important features such as localization. The exact structure of a bundle depends on whether you are creating an application, framework, or plug-in. It also depends on other factors such as the target platform and the type of plug-in.

The reason bundles and packages are sometimes considered to be interchangeable is that many types of bundles are also packages. For example, applications and loadable bundles are packages because they are usually treated as opaque directories by the system. However, not all bundles are packages and vice versa.

How the System Identifies Bundles and Packages

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

The Finder considers a directory to be a package if any of the following conditions are true:

- The directory has a known filename extension: `.app`, `.bundle`, `.framework`, `.plugin`, `.kext`, and so on.
- The directory has an extension that some other application claims represents a package type; see "Document Packages."
- The directory has its package bit set.

The preferred way to specify a package is to give the package directory a known filename extension. For the most part, Xcode takes care of this for you by providing templates that apply the correct extension. All you have to do is create an Xcode project of the appropriate type.

Most bundles are also packages. For example, applications and plug-ins are typically presented as a single file by the Finder. However, this is not true for all bundle types. In particular, a framework is a type of bundle that is treated as a single unit for the purposes of linking and runtime usage, but framework directories are transparent so that developers can view the header files and other resources they contain.

About Bundle Display Names

Display names give the user some control over how bundles and packages appear in the Finder without breaking clients that rely on them. Whereas a user can rename a file freely, renaming an application or framework might cause related code modules that refer to the application or framework by name to break. Therefore, when the user changes the name of a bundle, the change is superficial only. Rather than change the bundle name in the file system, the Finder associates a separate string (known as the **display name**) with the bundle and displays that string instead.

Display names are for presentation to the user only. You never use display names to open or access directories in your code, but you do use them when displaying the name of the directory to the user. By default, a bundle's display name is the same as the bundle name itself. However, the system may alter the default display name in the following cases:

- If the bundle is an application, the Finder hides the `.app` extension in most cases.
- If the bundle supports localized display names (and the user has not explicitly changed the bundle name), the Finder displays the name that matches the user's current language settings.

Although the Finder hides the `.app` extension for applications most of the time, it may display it to prevent confusion. For example, if the user changes the name of an application and the new name contains another filename extension, the Finder shows the `.app`. extension to make it clear that the bundle is an application. For example, if you were to add the `.mov` extension to the `Chess` application, the Finder would display `Chess.mov.app` to prevent users from thinking `Chess.mov` is a QuickTime file.

For more information about display names and specifying localized bundle names, see *File System Overview*.

The Advantages of Bundles

Bundles provide the following advantages for developers:

- Because bundles are directory hierarchies in the file system, a bundle just contains files. Therefore, you can use all of the same file-based interfaces to open your bundle resources as you do to open other types of files.
- The bundle directory structure makes it easy to support multiple localizations. You can easily add new localized resources or remove unwanted ones.
- Bundles can reside on volumes of many different formats, including multiple fork formats like HFS,

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

- Users can install, relocate, and remove bundles simply by dragging them around in the Finder.
- Bundles that are also packages, and are therefore treated as opaque files, are less susceptible to accidental user modifications, such as removal, modification, or renaming of critical resources.
- A bundle can support multiple chip architectures (PowerPC, Intel) and different address space requirements (32-bit/64-bit). It can also support the inclusion of specialized executables (for example, libraries optimized for a particular set of vector instructions).
- Most (but not all) executable code can be bundled. Applications, frameworks (shared libraries), and plug-ins all support the bundle model. Static libraries, dynamic libraries, shell scripts, and UNIX command line tools do not use the bundle structure.
- A bundled application can run directly from a server. No special shared libraries, extensions, and resources need to be installed on the local system.

Types of Bundles

Although all bundles support the same basic features, there are variations in the way you define and create bundles that define their intended usage:

- **Application** - An application bundle manages the code and resources associated with a launchable process. The exact structure of this bundle depends on the platform (iPhone OS or Mac OS X) that you are targeting. For information about the structure of application bundles, see "Application Bundles."
- **Frameworks** - A framework bundle manages a dynamic shared library and its associated resources, such as header files. An application can link against one or more frameworks to take advantage of the code they contain. For information about the structure of framework bundles, see "Anatomy of a Framework Bundle."
- **Plug-Ins** - Mac OS X supports plug-ins for many system features. Plug-ins are a way for an application to load custom code modules dynamically. The following list identifies some of the key types of plug-ins you might want to develop:
    - **Custom plug-ins** are plug-ins you define for your own purposes; see "Anatomy of a Loadable Bundle."
    - **Image Unit plug-ins** add custom image-processing behaviors to the Core Image technology; see *Image Unit Tutorial*.
    - **Interface Builder plug-ins** contain custom objects that you want to integrate into Interface Builder's library window; see *Interface Builder Plug-In Programming Guide*.
    - **Preference Pane plug-ins** define custom preferences that you want to integrate into the System Preferences application; see *Preference Pane Programming Guide*.
    - **Quartz Composer plug-ins** define custom patches for the Quartz Composer application; see *Quartz Composer Custom Patch Programming Guide*.
    - **Quick Look plug-ins** support the display of custom document types using Quick Look; see *Quick Look Programming Guide*.
    - **Spotlight plug-ins** support the indexing of custom document types so that those documents can be searched by the user; see *Spotlight Importer Programming Guide*.
    - **Sync Schema plug-ins** identify custom information that can be synchronized with the system; see *Sync Services Programming Guide*.

DOCKET ALARM Find authenticated court documents without watermarks at docketalarm.com.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.