Introduction

Applied to computer programs, resources are data files that accompany a program's executable code. Resources simplify the code you have to write by moving the creation of complex sets of data or graphical content outside of your code and into more appropriate tools. For example, rather than creating images pixel by pixel using code, it is much more efficient (and practical) to create them in an image editor. To take advantage of a resource, all your code has to do is load it at runtime and use it.

In addition to simplifying your code, resources are also an intimate part of the internationalization process for all applications. Rather than hard-coding strings and other user-visible content in your application, you can place that content in external resource files. Localizing your application then becomes a simple process of creating new versions of each resource file for each supported language. The bundle mechanism used in both Mac OS X and iPhone OS provides a way to organize localized resources and to facilitate the loading of resource files that match the user's preferred language.

This document provides information about the types of resources supported in Mac OS X and iPhone OS and how you use those resources in your code. This document does not focus on the resource-creation process. Most resources are created using either third-party applications or the developer tools provided in the `/Developer/Applications` directory. In addition, although this document refers to the use of resources in applications, the information also applies to other types of bundled executables, including frameworks and plug-ins.

Before reading this document, you should be familiar with the organizational structure imposed by application bundles. Understanding this structure makes it easier to organize and find the resource files your application uses. For information on the structure of bundles, see *Bundle Programming Guide*.

Organization of This Document

This document includes the following chapters:

- "About Resources" provides an introduction to the resource types supported in Mac OS X and iPhone OS.
- "Nib Files" describes the Cocoa-specific support for nib files.
- "Carbon Resources" describes the Carbon-specific support for nib files.
- "String Resources" describes the support for localized string resources in applications.
- "Image, Sound, and Video Resources" describes the support for image, sound, and video resources in applications.

See Also

The following ADC Reference Library documents are conceptually related to *Resource Programming Guide*:

- *Bundle Programming Guide* describes the bundle structure used by applications to store executable code and resources.
- *Internationalization Programming Topics* describes the process of preparing an application (and it's resources) for translation into other languages.
- *Interface Builder User Guide* describes the application used to create nib file resources.
- *Property List Programming Guide* describes the facilities in place for loading property-list resource files into a Cocoa application.
- *Property List Programming Topics for Core Foundation* describes the facilities in place for loading property-list resource files into a C-based application.

Last updated: 2009-01-06

About Resources

There are several reasons to use resources in your application:

- They can reduce the amount of code needed to create your application's user interface.
- They make it possible to change your application's user interface without changing any code.
- They make it easy to localize your application's user-visible content.
- They can store other types of custom data that might be difficult or time-consuming to create at runtime.

The following sections describe the types of resources typically found in applications and how you use them.

Nib Files

Nib files are the quintessential resource type used to create graphical applications. A nib file is a data archive describing the objects, configuration, and layout information associated with an application's user interface. You create nib files using the Interface Builder application, which provides a graphical assembly area for the windows and menus that comprise applications. You assemble windows and menus in Interface Builder by dragging and dropping custom views, controls, and other components from the provided library of objects. In addition to positioning items inside a window or view, you can configure the properties of those items as well so that they have the custom look and behavior you want for your application. In Cocoa applications, you can also create connections between objects to facilitate the passing of messages between them.

When you load a nib file, the nib-loading code recreates your objects exactly as they were when you designed them, including any custom configuration or connection information. Because the objects are given to you fully configured, it is possible to create complex user interfaces (see Figure 1-1) with very little code.

Figure 1-1 User interface resources

Nib files are supported widely by both Mac OS X and iPhone OS, although there are some subtle differences in how nib files are supported from environment to environment. Mac OS X and iPhone OS both provide the same basic level of support for loading nib files and using their contents. For example, both provide nib-loading support through the `NSBundle` class and automated support for loading the application's main nib file. In Mac OS X, Cocoa provides additional support for loading nib files associated with documents and for using the `NSNib` class to load nibs. The Carbon environment also supports the use of nib files, although the semantics for using them are different. For Carbon applications, nib files are a repository of user interface items that can be loaded one-by-one, instead of all at once.

Note: The term "nib" and the corresponding file extension are an acronym for "NeXT Interface Builder." The Interface Builder application was originally developed at NeXT Computer, whose OPENSTEP operating system was used as the basis for creating Mac OS X.

For general information on how to use nib files in Cocoa-based applications (including iPhone applications), see "Nib Files." For information on how to use nib files in Carbon applications, see "Carbon Resources." For information on how to create nib files, see *Interface Builder User Guide*.

String Resources

Text strings are a prominent part of most user interfaces. Text strings are commonly found in an application's nib files but may also be found in other places as well. For example, if an error occurs, an application might load a string corresponding to that error and display the string in an alert panel. Instead of hard-coding such strings inside source files, which would make localization much more difficult, an application can instead load them from a strings resource file.

A strings resource file (also known as a **strings file** because its file extension is `.strings`) is a human-readable text file (in the UTF-16 encoding) containing a set of string resources for an application. The purpose of strings files is to provide an external repository for an application's localizable text. An application can have any number of strings files and each strings file can contain any number of strings. Each entry in a strings file consists of a key-value pair where both the key and value are themselves strings. The key portion never changes and represents the identifier that your application uses to retrieve the string; however, the value for that key is typically translated to one of the languages your application supports.

Mac OS X provides tools to help you automatically generate strings files for your application. The tools search your code for any usage of specific string-loading routines and use that code to generate strings files for you. For more information about loading string resources and generating strings files, see "String Resources."

Image, Sound, and Multimedia Resources

Mac OS X and iPhone OS make extensive use of image resources (and to a lesser extent sound and multimedia resources) to create a unique visual style for the entire system. Some of these image resources are used to implement the glossy, three-dimensional texture commonly found in system components, such as the Aqua controls. Apple applications make extensive use of high-quality images to create the look and feel typically associated with the underlying system. Developers are similarly encouraged to use high-quality images to create beautiful and easy-to-use interfaces for their applications. The use of images can not only simplify your drawing code but for complex visual elements can improve performance by providing a prerendered version that can be cached and reused.

Because images are such an important part of graphical user interfaces in general, and Mac OS X and iPhone OS in particular, each system provides extensive support for loading and drawing image resource files. Both Mac OS X and iPhone OS provide support for loading and decoding image files saved in a variety of different formats. For resource files, however, the most commonly used formats include PNG, TIFF, PDF, GIF, and JPEG.

Just as you use images, you can use sound and multimedia resources to create a unique presentation style for your application. Although used less frequently than images, sounds can be used to provide feedback or to alert the user to special events. Audio support is provided by the Core Audio family of frameworks and also by

custom classes in the AppKit framework. Similarly, you can use movie clips to present video-based content. Mac OS X also provides extensive use for video and multimedia resources through the QuickTime and QuickTime Kit frameworks. In iPhone OS, similar support is provided by the Media Player framework.

For information about how to use image, sound, and video resources in your applications, see "Image, Sound, and Video Resources."

Property Lists

Property list files are a way to store custom configuration data outside of your application code. Mac OS X and iPhone OS use property lists extensively to implement features such as user preferences and information property list files for bundles. You can similarly use property lists to store private (or public) configuration data for your applications.

A property-list file is essentially a set of structured data values. You can create and edit property lists either programmatically or using the Property List Editor application (located in `/Developer/Applications/Utilities`). The structure of custom property-list files is completely up to you. You can use property lists to store string, number, Boolean, date, and raw data values. By default, a property list stores data in a single dictionary structure, but you can assign additional dictionaries and arrays as values to create a more hierarchical data set.

For information about using property lists, see *Property List Programming Guide* and *Property List Programming Topics for Core Foundation*.

Other Resource Files

In addition to the resource types listed in the preceding sections, Table 1-1 lists some additional resource file types you might find in an application bundle.

Table 1-1 Other resource types

Resource Type	Description
AppleScript files	In Mac OS X, AppleScript terminology and suite files contain information about the scriptability of an application. These files can use the file extensions `.sdef`, `.scriptSuite`, or `.scriptTerminology`. Because the actual AppleScript commands used to script an application are visible in user scripts and the Script Editor application, these resources need to be localized. For information on supporting AppleScript, see *AppleScript Overview*.
Help files	In Mac OS X, help content typically consists of a set of HTML files created using a standard text-editing program and registered with the Help Viewer application. (For information on how to register with Help Viewer, see *Apple Help Programming Guide*.) It is also possible to embed PDF files, RTF files, HTML files or other custom documents in your bundle and open them using

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.