Launch Services Programming Guide

Data Management: File Management

2009-11-17

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures and Tables

Chapter 1 **Launch Services Concepts 9**

Introduction

Launch Services is an API that enables a running application to open other applications or their document files or URLs (uniform resource locators) in a way similar to the Finder or the Dock. Using Launch Services, an application can perform such tasks as:

- Open (**launch** or activate) another application
- Open a document or a URL in another application
- Identify the preferred application for opening a given document or URL
- Register information about the kinds of document files and URLs an application is capable of opening
- Obtain appropriate information for displaying a file or URL on the screen, such as its icon, display name, and kind string
- Maintain and update the contents of the Recent Items menu

Although most of these services are normally performed by the Finder, other applications may also find them useful for purposes such as opening email attachments, following URLs embedded in a document, running helper applications, or opening embedded document components that were created by another application or require it for viewing or editing.

Many of Launch Services' capabilities were formerly provided by the Desktop Manager. With the advent of Mac OS X application bundles, however, the Desktop Manager has lost its usefulness, since it is not knowledgeable about bundled applications and simply ignores them. Similarly, Launch Services' facilities for dealing with URLs were formerly implemented through the Internet Config API. Launch Services replaces and supersedes the Desktop Manager and Internet Config with a new API providing similar functionality, but designed to operate properly in the Mac OS X environment.

Launch Services was created specifically to avoid the common need for applications to ask the Finder to open an application, document, or URL for them. In the past, opening such items in a way similar to the Finder required knowledge of several APIs, including the Desktop Manager, File Manager, Translation Manager, Internet Config, Process Manager, and Apple Event Manager. The Finder also had implicit knowledge of the desktop database and other information not available elsewhere for determining the correct application with which to open a given document.

Launch Services removes this specialized knowledge from the Finder and isolates it in a single, straightforward API available to any application. The Mac OS X Finder uses Launch Services to open applications, documents, and URLs at the user's request. Since the Finder does no additional processing beyond calling Launch Services, any client using Launch Services for these purposes is guaranteed to behave identically to the Finder itself.

Who Should Read This Document

This document is intended for all developers whose applications need to open other applications, open document files or URLs belonging to them, or display files or URLs on the screen in a manner similar to the Finder. For more detailed information on the Launch Services API, see the related document *Launch Services Reference*, which provides a comprehensive description of Launch Services functions, data types, constants, and result codes.

Organization of This Document

This document has the following chapters:

- "Launch Services Concepts" (page 9) presents the conceptual ideas underlying the Launch Services API, from the standpoint of both the developer and the user.
- "Launch Services Tasks" (page 25) tells how to use Launch Services to perform common tasks in your application.
- "Glossary" (page 31) defines various terms relating to Launch Services and its operations.

Launch Services Concepts

This chapter introduces both developers and users to basic information about Launch Services and its API.

Item Identification

In general, items to be operated on (such as applications, documents, or folders) can be identified to Launch Services in either of two ways:

- With a file-system reference (`FSRef`) designating a file residing on a local or remote file-system volume
- With a Core Foundation URL reference (`CFURL`) specifying a URL (uniform resource locator), typically (though not necessarily) denoting an item to be accessed via the Internet

Many Launch Services operations are implemented by pairs of related functions, one accepting a file-system reference as a parameter and the other a URL reference: for instance, the preferred application for opening an item can be found with either the `LSGetApplicationForItem` or the `LSGetApplicationForURL` function.

In addition, some Launch Services functions apply not to specific individual items but to families of items defined by certain identifying characteristics. These characteristics can include:

- A four-character **file type** code
- A four-character **creator signature**
- A filename extension
- A MIME (Multipurpose Internet Mail Extension) type

For instance, the Launch Services function `LSGetApplicationForInfo` finds the preferred application for a family of documents defined by their file type, creator signature, filename extension, or any combination of these characteristics; the `LSCopyApplicationForMIMEType` function finds the preferred application for items with a specified MIME type.

Note: In Mac OS X version 10.6 and later, Launch Services no longer considers file creator signatures when binding documents to applications. Launch Services ignores the creator signature when it's attached to a document. In addition, the functions `LSCopyKindStringForTypeInfo` and `LSGetApplicationForInfo` ignore the parameter containing the creator signature.

Item Information

Some Launch Services functions return requested information about an item or family of items. This can include:

- The item's file type
- The item's creator signature
- The item's filename extension
- The item's **display name:** the string to be used for displaying its name to the user (such as in the Finder or the Dock)
- The item's **kind string:** the string used (in the Finder's Get Info window or the Kind column of the Finder's list view, for instance) to characterize its general nature, such as `Application`, `Folder`, `Alias`, `JPEG Picture`, `QuickTime Movie`, or `FrameMaker Document`
- Flags describing various attributes of the item, including the following:
  - Is it a plain file (and not, for example, a directory, volume, or UNIX symbolic link)?
  - Is it an executable application?
  - If an application, can it run natively in Mac OS X?
  - If an application, does it require the Classic emulation environment?
  - If an application that can run either natively or in the Classic environment, does it prefer one environment or the other?
  - Is it a scriptable application?
  - Is it a container (such as a directory, package, or volume)?
  - Is it a packaged directory?
  - Is it the root directory of a volume?
  - Is it an alias?
  - Is it a UNIX symbolic link?
  - Is it invisible (that is, not displayed to the user in the Finder)?
  - Does it have a hidden filename extension?

Launch Services Database

Launch Services maintains a central data structure, the **Launch Services database,** in which it records all of the pertinent information about applications and the kinds of document files and URLs they are capable of opening. Whenever a new application becomes known to the system (such as when the user drags it from an installation disk into the Applications folder), the application is **registered** with Launch Services, which copies the needed information about the application into its database. Launch Services can then use this information to determine the preferred application for opening a given document file or URL.

Property List Keys for Launch Services

Launch Services obtains the information it needs about an application from the application's bundle information property list (`Info.plist`) or, in the case of a single-file application, from a `'plst'` resource in the application's resource fork. Table 1-1 shows the relevant keys. (See *Runtime Configuration Guidelines* for more detailed information on these and other property-list keys.)

Table 1-1 Property-list keys related to Launch Services

Key	Type	Description
`CFBundleDisplayName`	`String`	The application's display name
`CFBundleIconFile`	`String`	The name of the file containing the application's icon
`CFBundleIdentifier`	`String`	The application's bundle identifier
`CFBundleSignature`	`String`	The application's creator signature
`CFBundleDocumentTypes`	`Array`	An array of dictionaries describing the document types the application can open; see "Document Types" (page 13)
`CFBundleURLTypes`	`Array`	An array of dictionaries describing the URL types the application can open; see "URL Types" (page 13)
`LSRequiresCarbon`	`String`	Must the application run natively in Mac OS X?
`LSPrefersCarbon`	`String`	Does the application prefer to run natively in Mac OS X?
`LSRequiresClassic`	`String`	Must the application run in the Classic emulation environment?
`LSPrefersClassic`	`String`	Does the application prefer to run in the Classic emulation environment?
`LSBackgroundOnly`	`String`	Does the application run only in the background?
`LSUIElement`	`String`	Is the application a user interface element (that is, has no menu bar and should not appear in the Dock or the Force Quit window)?

The `CFBundleDisplayName` key specifies the application's display name; this key can be localized by including it in the `InfoPlist.strings` file of the appropriate `.lproj` subdirectory. `CFBundleIconFile` identifies the file containing the icon image to be used for displaying the application on the screen.

`CFBundleIdentifier` defines the application's **bundle identifier,** a unique identifying string used to locate its bundle at runtime. `CFBundleSignature` is the application's **creator signature,** a four-character code that identifies document files belonging to this application.

The keys `LSRequiresCarbon`, `LSPrefersCarbon`, `LSRequiresClassic`, `LSPrefersClassic`, `LSBackgroundOnly`, and `LSUIElement` specify various aspects of the environment in which the application should be run. A string value of `"1"` for any of these keys declares the corresponding attribute to be true. (In Mac OS X version 10.2 or later, these keys can also take values of type `Boolean` or `Number` rather than `String`, but such values are not supported in earlier system versions.)

The `LSRequiresCarbon`, `LSPrefersCarbon`, `LSRequiresClassic`, and `LSPrefersClassic` keys are mutually exclusive: at most one of them can be set to `"1"`. You can obtain information about the values of these four keys via the `flags` field of the item-information record returned by the Launch Services functions `LSCopyItemInfoForRef` and `LSCopyItemInfoForURL`. `LSRequiresCarbon` specifies that the application must be run natively in Mac OS X and cannot be run in the Classic emulation environment; `LSRequiresClassic` means the reverse. `LSPrefersCarbon` and `LSPrefersClassic` indicate that the application can run in either environment but has a preference for one or the other; the Finder offers the user the choice of which environment to use by displaying a checkbox in the application's Get Info window labeled "Open in the Classic environment," initially selected or deselected depending on which key the application itself specifies.

Note: The use of the word `Carbon` in the names `LSRequiresCarbon` and `LSPrefersCarbon` is misleading, since these keys actually signify that the application requires or prefers to run natively in MacOSX, irrespective of whether it is specifically Carbon-based; in particular, `LSRequiresCarbon` can apply equally well to Cocoa or Carbon applications. Note also the subtle difference in meaning between the `LSRequiresCarbon` flag in the bundle information property list and the `kLSItemInfoIsNativeApp` flag in the item-information record: the former indicates that the application *must* run natively and *cannot* run in the Classic environment, while the latter means only that the application *is capable of* running natively. Applications with `kLSItemInfoIsNativeApp` set may also be capable of running in the Classic environment, depending on the setting of the `kLSItemInfoPrefersNative` and `kLSItemInfoPrefersClassic` flags.

If none of the four keys is set to `"1"`, Launch Services infers the application's required or preferred environment by other means:

- If the application is bundled, it is assumed to be a native Mac OS X application and is run natively (equivalent to `LSRequiresCarbon`).
- If the application is not bundled and its resource fork contains a `'plst'` or `'carb'` resource, it is assumed to be compatible with either environment but prefer to run natively (equivalent to `LSPrefersCarbon`).
- If the application is not bundled and has no `'plst'` or `'carb'` resource, it is assumed to be a Classic application incapable of running natively (equivalent to `LSRequiresClassic`).

The most important property-list keys, for Launch Services' purposes, are `CFBundleDocumentTypes` and `CFBundleURLTypes`. The value associated with each of these keys is an array of dictionaries (**type-definition** and **scheme-definition dictionaries,** respectively), each of which declares (**claims**) a family of document files or URLs that the application is prepared to handle. Launch Services uses this information in deciding what application to use to open a given document file or URL.

Document Types

A type-definition dictionary defines a **document type,** a family of document files that the application can handle. Table 1-2 shows the relevant keys in this dictionary. (See *Runtime Configuration Guidelines* for more detailed information on these and other keys in the type-definition dictionary.)

Table 1-2 Keys in a type-definition dictionary

Key	Type	Description
`CFBundleTypeName`	`String`	The abstract name of the document type (also called its kind string)
`CFBundleTypeIconFile`	`String`	The name of the icon file for displaying documents of this type
`CFBundleTypeOSTypes`	`Array`	An array of four-character file types for documents belonging to this document type
`CFBundleTypeExtensions`	`Array`	An array of filename extensions for documents belonging to this document type
`CFBundleTypeMIMETypes`	`Array`	An array of MIME types for documents belonging to this document type
`CFBundleTypeRole`	`String`	The role the application claims with respect to documents of this type; see "Application Roles" (page 14)
`LSTypeIsPackage`	`Boolean`	Specifies whether the document is distributed as a bundle.

The `CFBundleTypeName` key specifies the document type's kind string, a user-visible description used to characterize documents of this type on the screen (such as in the Finder's Get Info window or in the Kind column of the Finder's list view). This key can be localized by including it in the `InfoPlist.strings` file of the appropriate `.lproj` subdirectory. `CFBundleTypeIconFile` identifies the file containing the icon image to be used for displaying documents of this type on the screen. `LSTypeIsPackage` specifies whether the document is a packaged bundle (`true`) or a single file (`false`).

Files belonging to a given document type may be characterized by their file types, filename extensions, or MIME types. The `CFBundleTypeOSTypes` key in the type-definition dictionary specifies an array of four-character file type codes that characterize documents of this type; similarly, `CFBundleTypeExtensions` specifies an array of filename extensions and `CFBundleTypeMIMETypes` an array of MIME types. Any of these individual keys can be omitted if the corresponding file characteristic is not relevant, but at least one of them must be present for the file type to be nonempty. To allow an application to accept files of unrestricted file type or extension during drag-and-drop operations, you can use the special wild-card values `'****'` or `'*'` for `CFBundleOSTypes` or `CFBundleTypeExtensions`, respectively. (These are honored only in drag-and-drop operations and not when the user opens a document by double-clicking.) Finally, the `CFBundleTypeRole` key specifies the role that the application claims with respect to documents of the given type, as described under "Application Roles" (page 14).

URL Types

A scheme-definition dictionary is similar to a type-definition dictionary, but defines a **URL type**—a family of URLs that the application can handle—rather than a document type. Table 1-3 shows the keys in this type of dictionary. (See *Runtime Configuration Guidelines* for more detailed information on these keys.)

Table 1-3 Keys in a scheme-definition dictionary

Key	Type	Description
`CFBundleURLName`	`String`	The abstract name of the URL type (also called its kind string)
`CFBundleURLIconFile`	`String`	The name of the icon file for displaying URLs of this type
`CFBundleURLSchemes`	`Array`	An array of URL schemes for URLs belonging to this URL type

The `CFBundleURLName` key specifies the URL type's kind string, a user-visible description used to characterize URLs of this type on the screen (such as in the Finder's Get Info window or in the Kind column of the Finder's list view). This key can be localized by including it in the `InfoPlist.strings` file of the appropriate `.lproj` subdirectory. `CFBundleURLIconFile` identifies the file containing the icon image to be used for displaying URLs of this type on the screen.

URLs belonging to a given URL type are characterized by their scheme components, such as `http`, `ftp`, `mailto`, or `file`. The `CFBundleURLSchemes` key in the scheme-definition dictionary specifies an array of schemes that characterize URLs of this type.

Application Roles

In declaring a document type, an application can claim a particular **role** with respect to documents of that type, defining the kinds of operations the application is capable of performing on such documents. The role is declared via the `CFBundleTypeRole` key in the application's type-definition dictionary for the given document type. Launch Services recognizes three such roles:

- `Editor`. The application can read, present, manipulate, and save documents of the given type.
- `Viewer`. The application can read and present documents of the given type, but cannot manipulate or save them.
- `None`. The application cannot operate on documents of the given type. This role is useful for declaring information about document types that the application is incapable of opening, such as their abstract names and icon files.

Launch Services defines a set of bit-mask constants of type `LSRolesMask` denoting the various possible roles. Launch Services functions that find the preferred application for a document or family of documents (`LSGetApplicationForItem`, `LSGetApplicationForURL`, and `LSGetApplicationForInfo`), or that determine whether a given application can open a designated document (`LSCanRefAcceptItem` and `LSCanURLAcceptURL`), take a parameter of this type to specify the application's desired role with respect to the document.

Application Registration

All applications available on the user's system must be **registered** to make them known to Launch Services and copy their document binding and other information into its database. It isn't ordinarily necessary to perform this task explicitly, since a variety of utilities and services built into the Mac OS X system software take care of it automatically:

- A built-in background tool, run whenever the system is booted or a new user logs in, automatically searches the Applications folders in the system, network, local, and user domains and registers any new applications it finds there. (This operation is analogous to "rebuilding the desktop" in earlier versions of Mac OS.)
- The Finder automatically registers all applications as it becomes aware of them, such as when they are dragged onto the user's disk or when the user navigates to a folder containing them.
- When the user attempts to open a document for which no preferred application can be found in the Launch Services database, the Finder presents a dialog asking the user to select an application with which to open the document. It then registers that application before launching it.

In spite of these automatic registration utilities, it may sometimes be necessary to register an application explicitly with Launch Services. For example, although developers are encouraged to package their applications so that they can be installed by simply dragging them onto the user's disk, some applications may require more elaborate custom installer software. In such cases, the installer should call one of the Launch Services registration functions `LSRegisterFSRef` or `LSRegisterURL` to register the application explicitly.

The registration functions take a Boolean parameter, `inUpdate`, which controls the function's behavior when the application being registered already exists in the Launch Services database. If this parameter is `true`, Launch Services unconditionally reregisters the application, replacing any previous information that may already exist for it in the database; if the parameter is `false`, the application is reregistered only if its current modification time is more recent than that recorded in the database.

After making any significant change in an application's Launch Services–related information, you should either reregister the application explicitly, by calling `LSRegisterFSRef` or `LSRegisterURL` with `inUpdate` set to `true`, or update the modification time of the application to ensure that it will be updated by the automatic registration utilities described above.

Note: You can update an application's modification time using the BSD `touch` command in a Terminal window. For example, the command `touch /Applications/TextEdit.app` sets the modification time of TextEdit to the current time.

Open Operations

The primary purpose of Launch Services is to open applications, documents, and URLs. Exactly how this is done depends on the kind of item to be opened, as described in the following sections.

Opening Applications

When the item to be opened is an application (or a URL with scheme `file` designating an application), Launch Services checks whether the application is already running and proceeds accordingly:

- If the application is not already running, Launch Services starts it up (**launches** it) and sends it an `'oapp'` ("open application") Apple event. The application should respond to this event by performing its normal startup processing.

- If the application is already running, Launch Services activates it (brings it to the front of the screen) and sends it an `'rapp'` ("reopen application") Apple event. This instructs the application to take some additional action, if necessary, to provide visual feedback to the user that it has become active: for example, it might open an empty document window or a document creation dialog if it has no other windows already open.

In either case, Launch Services adds the application to the Recent Items submenu in the Apple menu.

Opening Documents

If the item to be opened is a document (or a URL with scheme `file` designating a document file), Launch Services must first determine what application to use to open the item. This is known as the item's **preferred application.** As described under "User-Specified Binding Preferences" (page 17), the Mac OS X user interface allows the user to specify an explicit binding between a document and its preferred application. If such an explicit binding has been specified, it takes precedence over any other considerations; if not, Launch Services uses a set of implicit binding criteria to determine the preferred application, as described under "Preferred Applications" (page 22).

Once the preferred application has been determined, Launch Services launches or activates it (depending on whether it is already running) and sends it an `'odoc'` ("open document") Apple event instructing it to open the specified document. (If the document is to be printed rather than merely opened, a `'pdoc'` ("print document") Apple event is sent instead of `'odoc'`; in the case of a `file` URL, if the application claims to handle URLs with that scheme, it is sent a `'GURL'` ("get URL") Apple event instead.)

Finally, Launch Services adds both the application and the document (or URL) to the Recent Items submenus in the Apple menu.

Opening URLs

If the item to be opened is a URL with a scheme other than `file`, it is opened in essentially the same way as a document, but with the following exceptions:

- As described under "Preferred Applications" (page 22), the implicit binding criteria for selecting a preferred application are based on the URL's scheme component rather than on a creator signature, file type, or filename extension.
- The Apple event sent to the application after launching or activating it is `'GURL'` ("get URL") rather than `'odoc'` ("open document").

After the preferred application is launched or activated, Launch Services adds the application to the Recent Items submenu in the Apple menu.

Launch Options

When opening an application (whether by itself or to open one or more documents or URLs), you can specify various **launch options** to control the manner in which it is launched or activated. These can include:

- Whether to open the documents or URLs (if any) in a specifically designated application or in their own preferred applications

- Whether to print the documents or URLs (if any) or merely open them
- Whether to add the application and the documents (if any) to the Finder's Recent Items menu or to suppress this action
- Whether to permit the application to open in the background or to fail if it is a background-only application
- Whether to open the application without bringing it to the foreground
- Whether to permit the application to open in the Classic emulation environment or to fail if it is a Classic-only application
- Whether to launch a new instance of the application, even if another instance is already running
- Whether to hide the application after launching it
- Whether to hide all other applications after launching this one
- Whether to launch the application synchronously or asynchronously (see next section)

Synchronous and Asynchronous Launch

One of the options you can specify when launching an application is whether to launch it **synchronously** or **asynchronously:**

- In a synchronous launch, control does not return from the Launch Services function launching the application until the application has completed its launch sequence (indicated visually to the user when the application's icon stops "bouncing" in the Dock).
- In an asynchronous launch, control returns immediately, while the icon in the Dock is still "bouncing." When the launch sequence has been completed, your application is notified with a Carbon event of type `kEventAppLaunchNotification`. You can install an event handler callback routine to respond to this event in whatever way is appropriate. In your call to the Launch Services function that launches the application, you can supply an arbitrary 4-byte **reference constant** that will be passed back to your handler routine as part of the notification event.

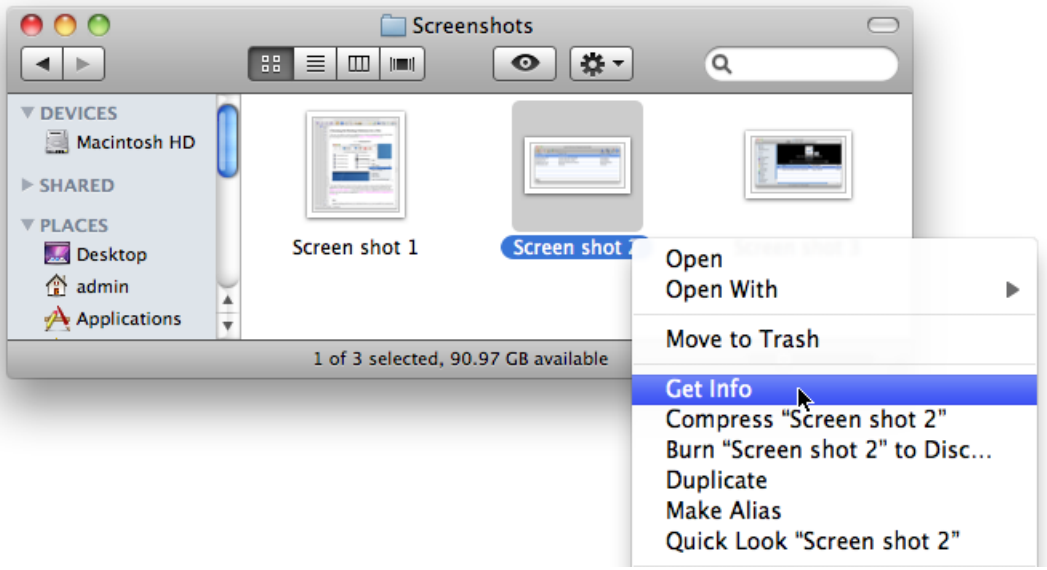
User-Specified Binding Preferences

Launch Services' first priority in determining the preferred application for a file or a non-file URL is whether the user has specified an explicit binding preference for that item.

Choosing the Binding Preference for a File

The user can specify a preferred application for a file by selecting the item in the Finder and choosing the Get Info command (see Figure 1-1).

Figure 1-1 Choosing Get Info



The Open With pane of the Get Info window contains a pop-up menu listing all known applications in the Launch Services database that claim to accept the selected item (see Figure 1-2). The user can then choose an application from the menu to become the item's preferred application (Figure 1-3).

Note: Explicit binding preferences for individual items are not user-specific but systemwide—that is, they continue to apply to the given item on that same computer, even if a different user logs in.

Figure 1-2 Get Info window

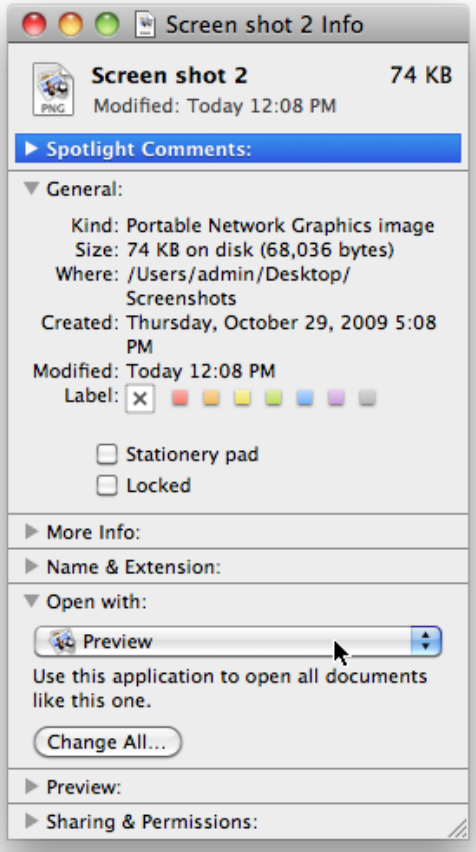
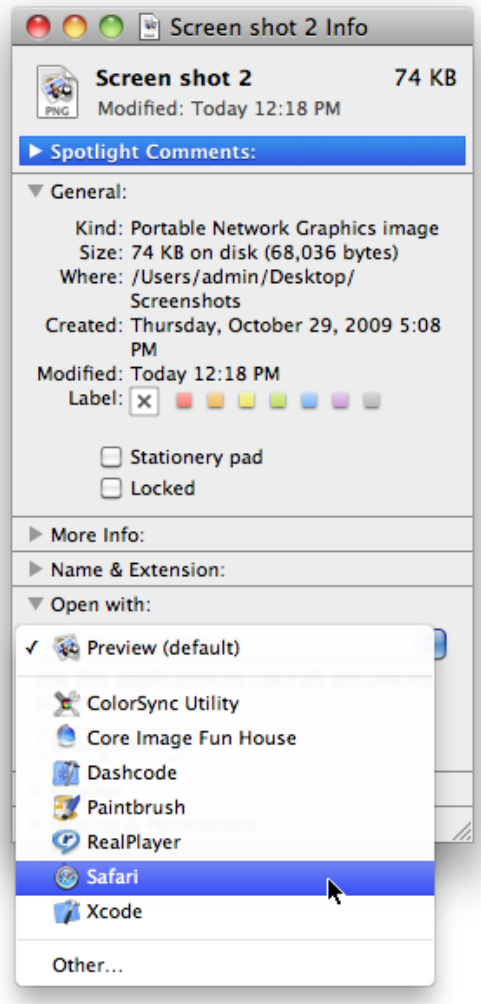
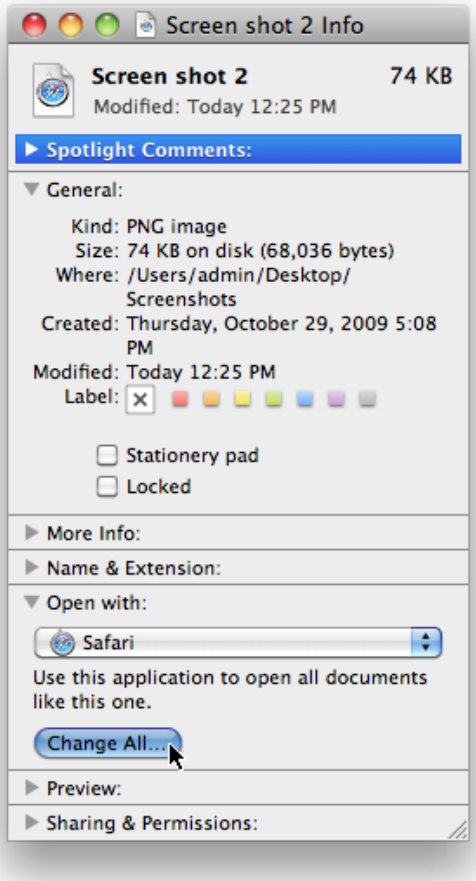


Figure 1-3 Selecting the preferred application for an item



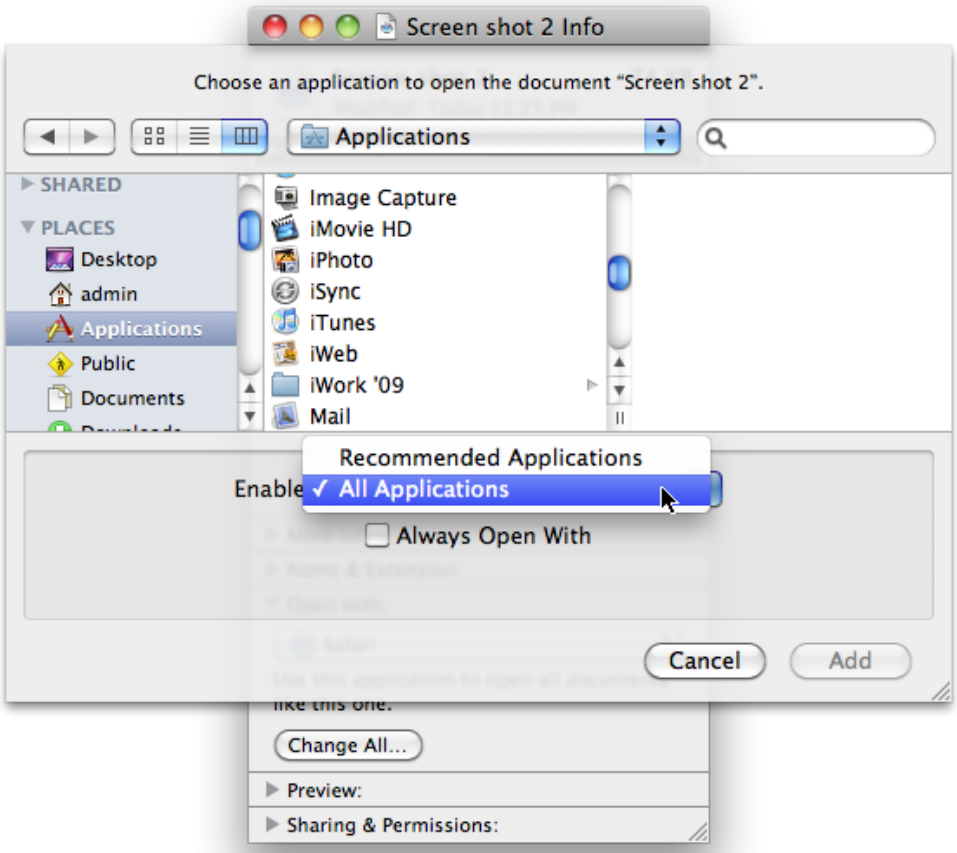
Clicking the Change All button (Figure 1-4) makes the chosen application the preferred application for all items of the same document or URL type, rather than just for the single item selected.

Figure 1-4 Selecting the preferred application for an item type



Occasionally, a user may wish to designate a preferred application that doesn't claim to accept a given document or URL. (This might be useful, for instance, for opening documents in a text-encoded format, such as HTML, as unencoded text in a text editor.) The Other item in the Open With pane's pop-up menu opens the dialog shown in Figure 1-5, in which the user can navigate to the desired application. The All Applications item in the pop-up menu labeled Show at the top of the dialog allows any desired application to be selected; Recommended Applications causes those not claiming to accept the item to be dimmed.

Figure 1-5 Choose Other Application dialog



Choosing the Binding Preference for a URL

There is no system-level user interface for setting non-file URL scheme handlers. However, individual applications can allow users to choose a preferred application for a specific URL scheme. For example:

- The Safari application allows users to set the `http:` handler by choosing a default web browser.
- The Mail application allows users to set the `mailto:` handler by choosing a default email reader.

Preferred Applications

Launch Services performs a document binding operation when it needs to:

- Compute the best icon to use for a document.
- Compute the kind string for a document, as shown in the Finder list view and some other contexts.

- Open a document.

Launch Services uses a series of prioritized binding criteria to determine the preferred application for opening a given document or URL. These are used by the Launch Services functions that open a document file (`LSOpenFSRef`, `LSOpenFromRefSpec`) or a URL (`LSOpenCFURLRef`, `LSOpenFromURLSpec`), as well as by those that merely locate the preferred application for such an item without actually opening it (`LSGetApplicationForItem`, `LSGetApplicationForURL`). They are also used by the `LSGetApplicationForInfo` function, which locates the preferred application for opening a family of items defined by specified identifying characteristics.

Preferred Application for a Document

For individual document files (whether specified by a file-system reference or a URL with scheme `file`), the criteria are as follows:

1. If the user has specified an explicit binding for the document (or for the entire document type to which it belongs), the preferred application is the one the user has specified.
2. If the document has a filename extension (or if one has been specified as a parameter to `LSGetApplicationForInfo`), find all applications in the Launch Services database that claim to accept documents with that extension.
3. If the document carries a four-character file type (or if one has been specified as a parameter), find all applications that claim to accept files of that type.
4. If more than one application has been found as a result of steps 2–3, apply the following criteria in the order shown:
    a. If the document carries a four-character creator signature (or if one has been specified as a parameter), give preference to any application that claims to accept documents with that signature (typically the application to which the signature belongs).
    b. Give preference to native OS X applications over those that run in the Classic emulation environment.
    c. Give preference to applications residing on the boot volume over those residing on other file-system volumes.
    d. Give preference to applications residing on a local volume over those residing on a remote volume.
    e. If two or more versions of the same application have been found, give preference to the one with the latest version number.

If two or more candidate applications remain after all of the foregoing criteria have been applied, Launch Services chooses one of the remaining applications in an unspecified manner.

Note: Criterion 4a does not apply in Mac OS X version 10.6 and later. Criteria 4c and 4d do not apply in Mac OS X version 10.2 and earlier. Apple reserves the right to change the selection criteria in future system releases.

Note: When the item to be opened is a file-system folder, it is treated as a document file whose preferred application is the Finder. This provides a convenient way of asking the Finder to open a window displaying the contents of a designated folder.

Preferred Application for a URL

The criteria for URLs with schemes other than `file` are similar to those for document files, except that the search is based on the URL's scheme rather than on file characteristics such as the creator signature, filename extension, and file type:

1. If the user has specified an explicit binding for the URL (or for the entire URL type to which it belongs), the preferred application is the one the user has specified.
2. If no explicit binding has been specified, find all applications in the Launch Services database that claim to accept URLs with the given scheme.
3. If more than one application has been found in step 2, apply the following criteria in the order shown:
    a. Give preference to native OS X applications over those that run in the Classic emulation environment.
    b. Give preference to applications residing on the boot volume over those residing on other file-system volumes.
    c. Give preference to applications residing on a local volume over those residing on a remote volume.
    d. If two or more versions of the same application have been found, give preference to the one with the latest version number.

If two or more candidate applications remain after all of the foregoing criteria have been applied, Launch Services chooses one of the remaining applications in an unspecified manner.

Note: Criteria 3b and 3c do not apply in Mac OS X version 10.2 and earlier. Apple reserves the right to change the selection criteria in future system releases.

Launch Services Tasks

This chapter summarizes how to use Launch Services to perform common tasks in your application.

Opening Items

The most common operation you'll want to perform with Launch Services is opening applications, document files, and URLs. Depending on the circumstances, you can use any of four Launch Services functions for this purpose: `LSOpenFSRef`, `LSOpenFromRefSpec`, `LSOpenCFURLRef`, or `LSOpenFromURLSpec`.

Opening Items by File-System Reference

When an item you wish to open is identified by a file-system reference (`FSRef`), the simplest way to open it is with `LSOpenFSRef`. You simply supply the file-system reference and Launch Services opens the item in a straightforward, no-frills, default way:

- If the designated item is an application:
  - If the application is not already running, it is launched and sent an `'oapp'` ("open application") Apple event.
  - If the application is already running, it is activated (brought to the front of the screen) and sent an `'rapp'` ("reopen application") Apple event.
- If the designated item is a document, its preferred application is launched (or activated if it was already running) and sent an `'odoc'` ("open document") Apple event instructing it to open the document.

`LSOpenFSRef` in turn calls the more general function `LSOpenFromRefSpec`, a "Swiss Army knife" function that provides access to the full range of options for opening applications and documents. You can call this function directly yourself if you need to request something other than the default behavior. For instance, you can use it to:

- Open more than one document at a time, in either the same or different applications
- Force a document to open in an application other than its own preferred application
- Open documents for printing rather than for simple viewing or editing
- Force an application to open in the Classic emulation environment
- Open a specified application and hide all others
- Prevent an application or document from being added to the Finder's Recent Items menu

Instead of a direct file-system reference to an item to be opened, you supply a pointer to a **launch specification,** a data structure of type `LSLaunchFSRefSpec` that identifies one or more items along with additional information about how to open them:

- To open one or more documents, pass an array of file-system references in the launch specification's `itemRefs` field; the `numDocs` field tells how many there are. If the `appRef` field is also non-null, it specifies the application in which to open the documents; otherwise, each document will be opened in its own preferred application.
- To open an application without specifying any documents, pass a file-system reference to the application in the launch specification's `appRef` field and set the `itemRefs` field to `NULL` and `numDocs` to `0`.

The additional information in the launch specification includes:

- A flag word (`launchFlags`) containing various launch options to control the manner in which the application is opened; see "Launch Options" (page 16)
- A pointer to an optional Apple event descriptor record (`passThruParams`) containing parameter information to be passed with the Apple event the application receives on opening
- An optional reference constant (`asyncRefCon`) to be passed to your Carbon event handler routine for asynchronous launch notifications, as described under "Synchronous and Asynchronous Launch" (page 17)

For both `LSOpenFSRef` and `LSOpenFromRefSpec`, the output parameter `outLaunchedRef` holds a pointer to a file-system reference that the function will set to indicate the application that was opened (or the first such application, in the case of multiple documents opened in different applications). If this information is not of interest, you can set this parameter to `NULL`.

Opening Items by URL

To open a URL, you use the Launch Services function `LSOpenCFURLRef` or `LSOpenFromURLSpec`. These are analogous to `LSOpenFSRef` and `LSOpenFromRefSpec`, but accept Core Foundation URL references (`CFURLRef`) instead of file-system references. These functions also are often useful when you have a file-system pathname to an application or document to be opened: you can construct a URL with scheme `file` containing the path and then use this URL in place of a file-system reference to open the item. The `LSOpenCFURLRef` and `LSOpenFromURLSpec` functions are the only way to open URLs with other schemes, such as `http`, `ftp`, or `mailto`.

Like `LSOpenFSRef` for file-system references, `LSOpenCFURLRef` opens a designated URL in its preferred application in the default way. The more general function `LSOpenFromURLSpec` accepts a launch specification (analogous to the one for `LSOpenFromRefSpec` but of type `LSLaunchURLSpec` rather than `LSLaunchFSRefSpec`) specifying in greater detail the manner in which the URL is to be opened. As with `LSOpenFromRefSpec`, you can call this function directly yourself if you need to request something other than the default behavior provided by `LSOpenCFURLRef`.

Both `LSOpenCFURLRef` and `LSOpenFromURLSpec` determine what application to use to open a specified URL, launch the application (or activate it if it is already running), and send it an Apple event instructing it to open the URL. (With `LSOpenFromURLSpec`, you can override the URL's preferred application by explicitly designating another application in the launch specification's `appURL` field.) Ordinarily, the application receives a `'GURL'` ("get URL") Apple event; but if the URL's scheme is `file` and the application doesn't claim to accept URLs with this scheme, it is sent an `'odoc'` ("open document") Apple event instead.

Like their counterparts for file-system references, both of the URL-based functions can optionally return information about which application was actually opened (or the first, in the case of multiple URLs opened in different applications). This information is passed back via a Core Foundation URL reference to which you supply a pointer in the output parameter `outLaunchedURL`. You can set this parameter to `NULL` if the identity of the application is not of interest.

Finding an Item's Preferred Application

To find the preferred application for a document file, URL, or MIME type without opening it, use the Launch Services function `LSGetApplicationForItem`, `LSGetApplicationForURL`, or `LSCopyApplicationForMIMEType`, respectively. You identify the item of interest with a file-system reference (`FSRef`) to the document, a Core Foundation URL reference (`CFURLRef`) to the URL, or a Core Foundation string reference (`CFStringRef`) to a string specifying the MIME type. Another Launch Services function, `LSGetApplicationForInfo`, finds the preferred application for a family of documents defined by their file type, creator signature, filename extension, or any combination of these characteristics.

In each case, you must supply a **role mask** (`LSRolesMask`) specifying one or more roles (`Editor`, `Viewer`, or `None`) that the application should claim with respect to the given item or family of items. (Note that `None` does not mean "no role at all," but rather refers to a specific role that the application can claim with respect to the item: that of providing identifying information such as a display name and icon file without actually being able to open the item itself.) If you don't care what role the application claims, use the mask value `kLSRolesAll`.

To receive the result, you pass a pointer to a file-system reference (in the `outAppRef` parameter), a Core Foundation URL reference (in the `outAppURL` parameter), or both; Launch Services will set the designated data structure to refer to the item's preferred application. You can pass a null pointer for either of these parameters if you don't care to receive the result in that form, but at least one of the two pointers must be non-null. (In the case of `LSCopyApplicationForMIMEType`, only the URL option is available; there is no `outAppRef` parameter.)

To find all known applications that can open a given item with a specified role, use the Launch Services function `LSCopyApplicationURLsForURL`. Although this function can only accept a URL reference and not a file-system reference, you can use it for document files as well, by passing a URL with scheme `file` referring to the desired document.

The Launch Services function `LSFindApplicationForInfo` locates an application based on its name, creator signature, bundle ID, or any combination of these characteristics. (Note that this differs from `LSGetApplicationForInfo` in that the specified characteristics apply to the application itself, rather than to the document files it can open.) As with other Launch Services functions discussed earlier, you can receive the result as either a file-system reference, a URL, or both, by passing pointers to the appropriate data structures to be filled with the information.

Testing Whether an Application Can Open an Item

Often it is useful to find out whether a given application claims the ability to open a particular document or URL. The Launch Services functions `LSCanRefAcceptItem` and `LSCanURLAcceptURL` provide this information. You supply either file-system references or URL references to the item and the target application, along with

a role mask and a flag word controlling certain technical aspects of the function's behavior; the function responds by setting a Boolean variable, to which you provide a pointer, to indicate whether the application can accept the designated item.

Registering an Application

It isn't ordinarily necessary to register an application explicitly with Launch Services, since this is done for you automatically whenever the application becomes known to the Finder, the system is booted, or a new user logs in (see "Application Registration" (page 14)). On those rare occasions when you do need to register an application explicitly (such as in a custom installer program), you can use the Launch Services function `LSRegisterFSRef` or `LSRegisterURL`, depending on whether the application is identified with a file-system reference or a Core Foundation URL reference. In either case, the application and its document binding information are copied into the Launch Services database, making the application available for opening documents and URLs.

Obtaining Information About an Item

You can use the Launch Services functions `LSCopyItemInfoForRef` and `LSCopyItemInfoForURL` to obtain a variety of information about file-system objects such as applications, documents, folders, or volumes. You supply a file-system reference or a Core Foundation URL reference (with scheme `file`) to identify the item of interest, along with a flag word (`LSRequestedInfo`) specifying the information you want and a pointer to an **item information record** (`LSItemInfoRecord`) in which to receive back the information. The information in this record can include the item's file type, creator signature, filename extension, and various flags describing attributes of the item (see "Item Information" (page 10)).

Two other pieces of information about an item that you may find useful are its display name (used for displaying its name to the user on the screen) and its kind string (used, for instance in the Finder's Get Info window or the Kind column of the Finder's list view, to characterize the item's general nature, such as `Application`, `Folder`, `Alias`, `JPEG Picture`, `QuickTime Movie`, or `FrameMaker Document`). You can obtain the display name with the Launch Services function `LSCopyDisplayNameForRef` or `LSCopyDisplayNameForURL` and the kind string with `LSCopyKindStringForRef`, `LSCopyKindStringForURL`, `LSCopyKindStringForTypeInfo`, or `LSCopyKindStringForMIMEType`.

Document Revision History

This table describes the changes to *Launch Services Programming Guide*.

Date	Notes
2009-11-17	In Mac OS X version 10.6 and later, Launch Services no longer considers file creator signatures when binding documents to applications.
2007-08-23	Made minor corrections. Changed title from "Launch Services Concepts and Tasks."
2003-12-01	New document that explains how to use Launch Services to open applications, documents, and URLs.

Document Revision History

Glossary

activate To bring a running application to the front of the screen, allowing the user to interact with it. Compare launch.

active extension A filename extensionclaimed by at least one application registered with Launch Services. Compare valid extension.

application An independently executable software program.

application bundle A bundle containing the executable code of an application and its associated resources.

application file A file containing the executable code of an application.

application package An application bundle presented to the user in the form of a package whose contents are ordinarily inaccessible for browsing.

asynchronous launch A launch operation in which control returns immediately to the calling program, without waiting for the launched application to complete its launch sequence. Compare synchronous launch.

binding information The information maintained in the Launch Services database about the kinds of documents and URLs an application is capable of opening.

binding preference A preference set by the user specifying the application in which to open a given document or URL.

binding rules The rules used by Launch Services to determine an item's default application according to the binding information in the Launch Services database.

bundle A directory containing executable code and related resources, structured according to conventions defined by Core Foundation Bundle Services.

bundle identifier A unique identifying string used to locate an application's bundle at runtime.

bundle information property list A collection of key-value pairs giving information about an application, stored in a file named `Info.plist` in its application bundle.

claim Said of an application, to declare to Launch Services that it is capable of opening documents or URLs of a given type.

Core Foundation URL reference A data object of type `CFURLRef` specifying a URL.

creator signature A four-character code associated with a file that identifies the application that created it or that should be used to open it.

default application The application selected by Launch Services, according to its own implicit binding rules, in which to open a given document or URL in the absence of an explicit binding preference set by the user.

directory A file-system object containing zero or more other named objects (files or other directories).

display name A string used for displaying an item's name to the user, such as in the Finder or the Dock.

document A unit or collection of data, contained in a file or package, that can be operated on by an application.

document file A file containing a document.

document package A package containing a document along with related resources.

document type A family of document files characterized by a given file type, creator signature, or filename extension. Compare URL type.

filename extension A string of characters at the end of a filename, preceded by a period (`.`), that characterizes the nature of the file or the structure of its contents.

file-system reference A data object of type `FSRef` designating a file residing on a local or remote file-system volume.

file type A four-character code associated with a file that characterizes its nature or the structure of its contents.

folder A directory presented to the user in such a way that its contents are accessible (subject to the appropriate permissions) for browsing. Compare package.

item Generically, an application, document, or URL to be operated on by Launch Services.

item information record A data structure of type `LSItemInfoRecord`, used by Launch Services to return information about an item.

kind string A string used (in the Finder's Get Info window, for example) to characterize the general nature of an item, such as `Application`, `Folder`, `Alias`, `JPEG Picture`, `QuickTime Movie`, or `FrameMaker Document`.

launch To start up an application that was not previously running. Compare activate.

launch options A set of flags specifying the manner in which an application is to be opened.

launch sequence The sequence of operations performed by an application immediately on being launched, indicated visually to the user by the application's icon "bouncing" in the Dock.

Launch Services A Mac OS X application programming interface that enables a running program to open other applications, documents, or URLs in a way similar to the Finder or the Dock.

Launch Services database The data structure in which Launch Services records information about available applications and the kinds of documents or URLs they are capable of opening.

launch specification A data structure of type `LSLaunchFSRefSpec` or `LSLaunchURLSpec`, used to specify to Launch Services the manner in which an item or items are to be opened.

MIME (Multipurpose Internet Mail Extension) A protocol used for adding attachments to email messages.

MIME type A string designating the type of data in an attachment transmitted via MIME, such as `text/plain`, `image/jpeg`, `audio/mp3`, or `video/quicktime`.

open Generically, to launch or activate an application or to present a document or URL for viewing or editing within an application.

package A directory presented to the user so that it appears to be a single file, and whose contents are ordinarily inaccessible for browsing by the user. Compare folder.

preferred application The application selected by Launch Services in which to open a given document or URL, either through an explicit binding preference set by the user or, in the absence of such a user preference, by applying Launch Services' own implicit binding rules for determining the item's default application.

reference constant An arbitrary data item available for use by a program to convey information for its own purposes in an operation or data structure.

register To make an application known to Launch Services, copying its binding information into the Launch Services database and making it available for opening documents and URLs.

role A characterization (such as `Editor` or `Viewer`) of the kinds of operations an application is capable of performing on documents or URLs of a given type.

role mask A parameter specifying the role or roles that an application should claim with respect to a given item in order to be considered a candidate for opening that item.

scheme The component of a uniform resource locator (URL) that identifies the type of resource it represents or the protocol to be used for accessing it, such as `http`, `ftp`, `mailto`, or `file`.

scheme-definition dictionary A dictionary, specified in an application's bundle information property list, that declares a particular URL type that the application claims to handle. Compare type-definition dictionary.

synchronous launch A launch operation in which control does not return to the calling program until the launched application has completed its launch sequence. Compare asynchronous launch.

type-definition dictionary A dictionary, specified in an application's bundle information property list, that declares a particular document type that the application claims to handle. Compare scheme-definition dictionary.

uniform resource locator A string, in a standard format, designating a file, Web page, or other resource, typically (but not necessarily) to be accessed via the Internet. Often used loosely in the context of Launch Services to refer to the resource so designated.

URL See uniform resource locator.

URL type A family of URLs characterized by a given scheme component. Compare document type.

valid extension A filename extension that does not contain spaces, periods, or characters that are not supported by the underlying file system. Compare active extension.