Launch Services Reference

Carbon > File Management

2006-07-13

Apple Inc.
© 2003, 2006 Apple Computer, Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple, the Apple logo, AppleScript, Carbon, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

Finder is a trademark of Apple Inc.

Adobe, Acrobat, and PostScript are trademarks or registered trademarks of Adobe Systems Incorporated in the U.S. and/or other countries.

UNIX is a registered trademark of The Open Group

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Launch Services Reference

Framework:	ApplicationServices/ApplicationServices.h
Companion guide	Launch Services Programming Guide
Declared in	LSInfo.h LSOpen.h

Overview

Mac OS X Launch Services is an API that enables a running application to open other applications or their document files 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 (uniform resource locator) 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.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

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 itself 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.

Before reading this document, you should be familiar with the related document, *Launch Services Programming Guide*, which presents a conceptual overview of Launch Services and its operations.

Functions by Task

This section describes the functions defined in the Launch Services API.

Locating an Application

The functions described in this section locate the preferred application for opening a given item or family of items or the application matching a given set of defining characteristics, or test whether an application can open a designated item.

`LSGetApplicationForItem` (page 25)
: Locates the preferred application for opening an item designated by file-system reference.

`LSGetApplicationForURL` (page 26)
: Locates the preferred application for opening an item designated by URL.

`LSGetApplicationForInfo` (page 24)
: Locates the preferred application for opening items with a specified file type, creator signature, filename extension, or any combination of these characteristics.

`LSCopyApplicationForMIMEType` (page 12)
: Locates the preferred application for opening items with a specified MIME type.

`LSCopyApplicationURLsForURL` (page 13)
: Locates all known applications suitable for opening an item designated by URL.

`LSCanRefAcceptItem` (page 9)
: Tests whether an application can accept (open) an item designated by file-system reference.

`LSCanURLAcceptURL` (page 10)
: Tests whether an application can accept (open) an item designated by URL.

`LSFindApplicationForInfo` (page 23)
: Locates an application with a specified creator signature, bundle ID, filename, or any combination of these characteristics.

Opening Items

The functions described in this section open a designated item or collection of items, or launch or activate a designated application.

`LSOpenApplication` (page 28)
: Launches the specified application.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`LSOpenItemsWithRole` (page 34)
Opens items specified as an array of values of type `FSRef` with a specified role.

`LSOpenURLsWithRole` (page 36)
Opens one or more URLs with the specified roles.

`LSOpenFSRef` (page 34)
Opens an item designated by file-system reference, in the default manner in its preferred application.

`LSOpenFromRefSpec` (page 31)
Opens one or more items designated by file-system reference, in either their preferred applications or a designated application.

`LSOpenCFURLRef` (page 29)
Opens an item designated by URL, in the default manner in its preferred application.

`LSOpenFromURLSpec` (page 32)
Opens one or more items designated by URL, in either their preferred applications or a designated application.

Obtaining Information About an Item

The functions described in this section obtain requested information about an item.

`LSCopyItemInfoForRef` (page 18)
Obtains requested information about an item designated by file-system reference.

`LSCopyItemInfoForURL` (page 19)
Obtains requested information about an item designated by URL.

`LSCopyDisplayNameForRef` (page 15)
Obtains the display name for an item designated by file-system reference.

`LSCopyDisplayNameForURL` (page 16)
Obtains the display name for an item designated by URL.

`LSCopyKindStringForRef` (page 20)
Obtains the kind string for an item designated by file-system reference.

`LSCopyKindStringForURL` (page 22)
Obtains the kind string for an item designated by URL.

`LSCopyKindStringForTypeInfo` (page 21)
Obtains a kind string for items with a specified file type, creator signature, filename extension, or any combination of these characteristics.

`LSCopyKindStringForMIMEType` (page 19)
Obtains the kind string for a specified MIME type.

`LSCopyItemAttribute` (page 16)
Obtains the value of an item's attribute.

`LSCopyItemAttributes` (page 17)
Obtains multiple item attribute values as a dictionary.

Getting and Setting Filename Extension Information

The functions described in this section obtain information about an item's filename extension, or control whether the extension should be hidden or shown on the screen.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`LSGetExtensionInfo` (page 27)
Obtains the starting index of the extension within a filename.

`LSSetExtensionHiddenForRef` (page 40)
Specifies whether the filename extension for an item designated by file-system reference should be hidden or shown.

`LSSetExtensionHiddenForURL` (page 40)
Specifies whether the filename extension for an item designated by URL should be hidden or shown.

Registering an Application

The functions described in this section register an application in the Launch Services database.

`LSRegisterFSRef` (page 37)
Registers an application, designated by file-system reference, in the Launch Services database.

`LSRegisterURL` (page 38)
Registers an application, designated by URL, in the Launch Services database.

Working With Role Handlers

The functions described in this section get and set application bundle identifiers for handlers of specified content types and URL schemes.

`LSCopyAllRoleHandlersForContentType` (page 11)
Returns an array of application bundle identifiers for applications capable of handling a specified content type with the specified roles.

`LSCopyDefaultRoleHandlerForContentType` (page 14)
Returns the application bundle identifier of the user's preferred default handler for the specified content type with the specified role.

`LSSetDefaultRoleHandlerForContentType` (page 39)
Sets the user's preferred default handler for the specified content type in the specified roles.

`LSGetHandlerOptionsForContentType` (page 28)
Gets the handler options for the specified content type.

`LSSetHandlerOptionsForContentType` (page 41)
Sets the handler option for the specified content type.

`LSCopyAllHandlersForURLScheme` (page 11)
Returns an array of application bundle identifiers for applications capable of handling the specified URL scheme.

`LSCopyDefaultHandlerForURLScheme` (page 13)
Returns the application bundle identifier of the user's preferred default handler for the specified URL scheme.

`LSSetDefaultHandlerForURLScheme` (page 38)
Sets the user's preferred default handler for the specified URL scheme.

Functions No Longer Used

The functions described in this section are no longer used.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`LSInit` (page 61) Deprecated in Mac OS X v10.3
(Deprecated. Formerly used to initialize Launch Services; now does nothing.)

`LSTerm` (page 61) Deprecated in Mac OS X v10.3
(Deprecated. Formerly used to terminate Launch Services; now does nothing.)

Functions

LSCanRefAcceptItem

Tests whether an application can accept (open) an item designated by file-system reference.

```
OSStatus LSCanRefAcceptItem (
   const FSRef *inItemFSRef,
   const FSRef *inTargetRef,
   LSRolesMask inRoleMask,
   LSAcceptanceFlags inFlags,
   Boolean *outAcceptsItem
);
```

Parameters

inItemFSRef
A pointer to a file-system reference designating the source item (the item to test for acceptance by the target application); see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

inTargetFSRef
A pointer to a file-system reference designating the target application; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

inRolesMask
A bit mask specifying the target application's desired role or roles with respect to the source item; see "Roles Mask" (page 46) for a description of this mask. If the role is unimportant, pass `kLSRolesAll`.

inFlags
Flags specifying behavior to observe during the acceptance test; see "Acceptance Flags" (page 54) for a description of these flags.

outAcceptsItem
A pointer to a Boolean value that, on return, will indicate whether the target application can accept the source item with at least one of the specified roles.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTCarbonShell

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Declared In
`LSInfo.h`

LSCanURLAcceptURL

Tests whether an application can accept (open) an item designated by URL.

```
OSStatus LSCanURLAcceptURL (
   CFURLRef inItemURL,
   CFURLRef inTargetURL,
   LSRolesMask inRoleMask,
   LSAcceptanceFlags inFlags,
   Boolean *outAcceptsItem
);
```

Parameters

inItemURL

A Core Foundation URL reference designating the source item (the item to test for acceptance by the target application); see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type.

inTargetURL

A Core Foundation URL reference designating the target application; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to an application file or application bundle.

inRolesMask

A bit mask specifying the target application's desired role or roles with respect to the source item; see "Roles Mask" (page 46) for a description of this mask. This parameter applies only to URLs with a scheme component of `file`, and is ignored for all other schemes. If the role is unimportant, pass `kLSRolesAll`.

inFlags

Flags specifying behavior to observe during the acceptance test; see "Acceptance Flags" (page 54) for a description of these flags.

outAcceptsItem

A pointer to a Boolean value that, on return, will indicate whether the target application can accept the source item with at least one of the specified roles.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
If the item URL's scheme is `file` (designating either a file or a directory), the acceptance test is based on the designated item's filename extension, file type, and creator signature, along with the role specified by the `inRolesMask` parameter; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`).

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSInfo.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSCopyAllHandlersForURLScheme

Returns an array of application bundle identifiers for applications capable of handling the specified URL scheme.

```
CFArrayRef LSCopyAllHandlersForURLScheme (
   CFStringRef inURLScheme
);
```

Parameters

inURLScheme

The URL scheme for which the application bundle identifiers are to be returned.

Return Value

An array containing the application bundle identifiers for applications capable of handling the URL scheme specified by `inURLScheme`, or `NULL` if no handlers are available.

Discussion

This function returns all of the application bundle identifiers that are capable of handling the specified URL scheme.

URL handling capability is determined according to the value of the `CFBundleURLTypes` key in an application's `Info.plist`. For information on the `CFBundleURLTypes` key, see the section "CFBundleURLTypes" in *Mac OS X Runtime Configuration Guidelines*.

Version Notes

Thread-safe since Mac OS X v10.4.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`LSInfo.h`

LSCopyAllRoleHandlersForContentType

Returns an array of application bundle identifiers for applications capable of handling a specified content type with the specified roles.

```
CFArrayRef LSCopyAllRoleHandlersForContentType (
   CFStringRef inContentType,
   LSRolesMask inRole
);
```

Parameters

inContentType

The content type. The content type is a uniform type identifier (UTI).

inRole

The role. Pass `kLSRolesAll` if any role is acceptable. For additional possible values, see "Roles Mask" (page 46).

Return Value

The application bundle identifiers for applications capable of handling the specified content type in the specified roles, or `NULL` if no handlers are available.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Discussion
This function returns all of the application bundle identifiers that are capable of handling the specified content type in the specified roles.

The `CFBundleDocumentTypes` key in an application's `Info.plist` can be used to set an application's content handling capabilities. The `LSItemContentTypes` key is particularly useful because it supports the use of UTIs in document claims. For information on the `CFBundleDocumentTypes` key, see the section "CFBundleDocumentTypes" in *Mac OS X Runtime Configuration Guidelines*.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSCopyApplicationForMIMEType

Locates the preferred application for opening items with a specified MIME type.

```
OSStatus LSCopyApplicationForMIMEType (
   CFStringRef inMIMEType,
   LSRolesMask inRoleMask,
   CFURLRef *outAppURL
);
```

Parameters

inMIMEType

A Core Foundation string object specifying the MIME type to consider; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. Comparison of MIME types is case-insensitive.

inRolesMask

A bit mask specifying the application's desired role or roles with respect to items with the specified MIME type; see "Roles Mask" (page 46) for a description of this mask. If the role is unimportant, pass `kLSRolesAll`.

outAppURL

A pointer to a Core Foundation URL reference that, on return, will identify the preferred application for items with the specified MIME type; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. You are responsible for releasing the URL reference object.

Return Value
A result code; see "Launch Services Result Codes" (page 57). If no application suitable for opening items with the specified MIME type is found in the Launch Services database, the function will return the result code `kLSApplicationNotFoundErr`.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.2 and later.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Declared In
`LSInfo.h`

LSCopyApplicationURLsForURL

Locates all known applications suitable for opening an item designated by URL.

```
CFArrayRef LSCopyApplicationURLsForURL (
   CFURLRef inURL,
   LSRolesMask inRoleMask
);
```

Parameters

inURL

A Core Foundation URL reference designating the item for which all suitable applications are requested; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type.

inRolesMask

A bit mask specifying the applications' desired role or roles with respect to the designated item; see "Roles Mask" (page 46) for a description of this mask. This parameter applies only to URLs with a scheme component of `file`, and is ignored for all other schemes. If the role is unimportant, pass `kLSRolesAll`.

Return Value
An array of Core Foundation URL references, one for each application that can open the designated item with at least one of the specified roles. You are responsible for releasing the array object. If no suitable applications are found in the Launch Services database, the function will return `NULL`

Discussion
If the item URL's scheme is `file` (designating either a file or a directory), the selection of suitable applications is based on the designated item's filename extension, file type, and creator signature, along with the role specified by the `inRolesMask` parameter; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`).

Version Notes
Thread-safe since Mac OS version 10.3.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`LSInfo.h`

LSCopyDefaultHandlerForURLScheme

Returns the application bundle identifier of the user's preferred default handler for the specified URL scheme.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
CFStringRef LSCopyDefaultHandlerForURLScheme (
   CFStringRef inURLScheme
);
```

Parameters

inURLScheme

The URL scheme for which the application bundle identifier is to be returned.

Return Value

The application bundle identifier of the specified URL scheme.

Discussion

This function returns the user's currently preferred default handler for the specified URL scheme.

URL handling capability is determined according to the value of the `CFBundleURLTypes` key in an application's `Info.plist`. For information on the `CFBundleURLTypes` key, see the section "CFBundleURLTypes" in *Mac OS X Runtime Configuration Guidelines*.

Version Notes

Thread-safe since Mac OS X v10.4.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`LSInfo.h`

LSCopyDefaultRoleHandlerForContentType

Returns the application bundle identifier of the user's preferred default handler for the specified content type with the specified role.

```
CFStringRef LSCopyDefaultRoleHandlerForContentType (
   CFStringRef inContentType,
   LSRolesMask inRole
);
```

Parameters

inContentType

The content type. The content type is a uniform type identifier (UTI).

inRole

The role. Pass `kLSRolesAll` if any role is acceptable. For additional possible values, see "Roles Mask" (page 46).

Return Value

The application bundle identifier of the default handler for the specified content type in the specified roles, or `NULL` if no handler is available.

Discussion

This function returns the user's currently preferred default handler for the specified content type. Say, for example, that `LSSetDefaultRoleHandlerForContentType` (page 39) has been used to set "com.Apple.TextEdit" for the "public.xml" content type. When a file whose content type is "public.xml" is double-clicked, TextEdit will be launched to open the file. If you call `LSCopyDefaultRoleHandlerForContentType (CFSTR("public.xml"), kLSRolesAll)`, the string `com.apple.TextEdit` is returned.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

The `CFBundleDocumentTypes` key in an application's `Info.plist` can be used to set an application's content handling capabilities. The `LSItemContentTypes` key is particularly useful because it supports the use of UTIs in document claims. For information on the `CFBundleDocumentTypes` key, see the section "CFBundleDocumentTypes" in *Mac OS X Runtime Configuration Guidelines*.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSCopyDisplayNameForRef

Obtains the display name for an item designated by file-system reference.

```
OSStatus LSCopyDisplayNameForRef (
   const FSRef *inRef,
   CFStringRef *outDisplayName
);
```

Parameters

`inRef`

A pointer to a file-system reference designating the item whose display name is requested; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

`outDisplayName`

A pointer to a Core Foundation string object that, on return, will contain the item's display name; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The item's display name is returned in the form in which it will appear on the user's screen; it may be localized (for applications and folders), and it excludes the filename extension if the extension is set to be hidden and the Finder preference to always show extensions is not enabled.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
QTMetaData

Declared In
`LSInfo.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSCopyDisplayNameForURL

Obtains the display name for an item designated by URL.

```
OSStatus LSCopyDisplayNameForURL (
   CFURLRef inURL,
   CFStringRef *outDisplayName
);
```

Parameters

inFileURL

A Core Foundation URL reference designating the item whose display name is requested; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to either a file or a directory.

outDisplayName

A pointer to a Core Foundation string object that, on return, will contain the item's display name; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Discussion

The item's display name is returned in the form in which it will appear on the user's screen; it may be localized (for applications and folders), and it excludes the filename extension if the extension is set to be hidden and the Finder preference to always show extensions is not enabled.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`LSInfo.h`

LSCopyItemAttribute

Obtains the value of an item's attribute.

```
OSStatus LSCopyItemAttribute (
   const FSRef *inItem,
   LSRolesMask inRoles,
   CFStringRef inAttributeName,
   CFTypeRef *outValue
);
```

Parameters

inItem

The `FSRef` of the item to query.

inRoles

The roles. When obtaining attributes related to document binding (such as `kLSItemRoleHandlerDisplayName`), at least one of the roles must be provided by the application selected. Pass `kLSRolesAll` if any role is acceptable.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

inAttributeName

The name of the attribute to copy. For possible values, see "Item Attribute Constants" (page 51).

outValue

A pointer to a `CFTypeRef`. On return, the `CFTypeRef` is set to the copied attribute value (a CF object), or is `NULL` if an error occurs. The type of the returned object varies depending on the attribute that is requested.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSCopyItemAttributes

Obtains multiple item attribute values as a dictionary.

```
OSStatus LSCopyItemAttributes (
   const FSRef *inItem,
   LSRolesMask inRoles,
   CFArrayRef inAttributeNames,
   CFDictionaryRef *outValues
);
```

Parameters

inItem

The `FSRef` of the item to query.

inRoles

The roles. When obtaining attributes related to document binding (such as `kLSItemRoleHandlerDisplayName`), at least one of the roles must be provided by the application selected. Pass `kLSRolesAll` if any role is acceptable.

inAttributeNames

A `CFArrayRef` for an array containing the attribute names to copy. For possible values, see "Item Attribute Constants" (page 51).

outValues

On return, a pointer a `CFDictionaryRef` for a dictionary whose keys are the attribute names specified by the `inAttributeNames` parameter and whose values are the attribute's values. The `CFTypeID` of each value in the dictionary varies by attribute. See "Item Attribute Constants" (page 51) for the data type of each value. If the item does not have a specified attribute, the key for the attribute is not in the dictionary.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Version Notes
Thread-safe since Mac OS X v10.4.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSCopyItemInfoForRef

Obtains requested information about an item designated by file-system reference.

```
OSStatus LSCopyItemInfoForRef (
   const FSRef *inItemRef,
   LSRequestedInfo inWhichInfo,
   LSItemInfoRecord *outItemInfo
);
```

Parameters

inItemRef

A pointer to a file-system reference designating the item about which information is requested; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

inWhichInfo

Flags specifying what information to obtain; see "Requested-Information Flags" (page 50) for a description of these flags.

outItemInfo

A pointer to an item-information record that, on return, will contain the requested information; see LSItemInfoRecord (page 45) for a description of this structure.

If you request the item's filename extension (field `extension` of the item-information record, requested by flag `kLSRequestExtension`), you are responsible for releasing the Core Foundation string object in which the extension is returned.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The information obtained about an item can include its filename extension, file type, creator signature, and various item-information flags (indicating, for example, whether the item is an application, or whether it has a hidden extension); see "Item-Information Flags" (page 52) for a description of these flags.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
QTCarbonShell

Declared In
`LSInfo.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSCopyItemInfoForURL

Obtains requested information about an item designated by URL.

```
OSStatus LSCopyItemInfoForURL (
   CFURLRef inURL,
   LSRequestedInfo inWhichInfo,
   LSItemInfoRecord *outItemInfo
);
```

Parameters

inFileURL

A Core Foundation URL reference designating the item about which information is requested; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to either a file or a directory.

inWhichInfo

Flags specifying what information to obtain; see "Requested-Information Flags" (page 50) for a description of these flags.

outItemInfo

A pointer to an item-information record that, on return, will contain the requested information; see LSItemInfoRecord (page 45) for a description of this structure.

If you request the item's filename extension (field `extension` of the item-information record, requested by flag `kLSRequestExtension`), you are responsible for releasing the Core Foundation string object in which the extension is returned.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Discussion

The information obtained about an item can include its filename extension, file type, creator signature, and various item-information flags (indicating, for example, whether the item is an application, or whether it has a hidden extension); see "Item-Information Flags" (page 52) for a description of these flags.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CarbonCocoa_PictureCursor

Declared In

`LSInfo.h`

LSCopyKindStringForMIMEType

Obtains the kind string for a specified MIME type.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
OSStatus LSCopyKindStringForMIMEType (
   CFStringRef inMIMEType,
   CFStringRef *outKindString
);
```

Parameters

inMIMEType

A Core Foundation string object specifying the MIME type whose kind string is requested; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. Comparison of MIME types is case-insensitive.

outKindString

A pointer to a Core Foundation string object that, on return, will contain the kind string for the specified MIME type; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The kind string (which may be localized) is obtained from the preferred application for opening items of the specified the MIME type, if one is found in the Launch Services database; otherwise, a more generic kind string is chosen.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`LSInfo.h`

LSCopyKindStringForRef

Obtains the kind string for an item designated by file-system reference.

```
OSStatus LSCopyKindStringForRef (
   const FSRef *inFSRef,
   CFStringRef *outKindString
);
```

Parameters

inFSRef

A pointer to a file-system reference designating the item whose kind string is requested; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

outKindString

A pointer to a Core Foundation string object that, on return, will contain the item's kind string; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Discussion
The kind string (which may be localized) is obtained from the item's preferred application, if one is found in the Launch Services database; otherwise, a more generic kind string is chosen. For example, the kind string might be `FrameMaker Document`, or just `Document` if the item is a document for which no application is found.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSInfo.h`

LSCopyKindStringForTypeInfo

Obtains a kind string for items with a specified file type, creator signature, filename extension, or any combination of these characteristics.

```
OSStatus LSCopyKindStringForTypeInfo (
   OSType inType,
   OSType inCreator,
   CFStringRef inExtension,
   CFStringRef *outKindString
);
```

Parameters

inType

The file type to consider. Comparison of file types is case-sensitive. Pass `kLSUnknownType` if the items' file type is unimportant.

inCreator

The creator signature to consider. Comparison of creator signatures is case-sensitive. Pass `kLSUnknownCreator` if the items' creator signature is unimportant.

inExtension

A Core Foundation string object specifying the filename extension to consider; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. Comparison of filename extensions is case-insensitive. Pass `NULL` if the items' filename extension is unimportant.

outKindString

A pointer to a Core Foundation string object that, on return, will contain the requested kind string; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
This function obtains the kind string that most closely describes items having the specified characteristics. It is useful when you want to display the kind string for a document you do not yet have (such as an email attachment).

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

You can request any combination of one, two, or all three of the characteristics specified by the *inType*, *inCreator*, and *inExtension* parameters; at least one of these characteristics must be supplied. The kind string (which may be localized) is obtained from the preferred application for opening such items, if one is found in the Launch Services database; otherwise, a more generic kind string is chosen. For example, the kind string might be `FrameMaker Document`, or just `Document` if no suitable application is found.

Note that since the choice of a preferred application is subject to any document binding preferences the user may have set, the kind string will not necessarily be obtained from the default application that matches the specified creator signature (if any), but may instead be taken from a user-specified application that overrides the default. For example, if the user has specified that files of type `'PDF '` and creator `'ACRO'` should be opened in the Preview application rather than in Acrobat, the kind string for this combination of characteristics will be that defined for `'PDF '` files by Preview and not by Acrobat.

Version Notes
Thread-safe since Mac OS version 10.2

Availability
Available in Mac OS X v10.2 and later.

Declared In
`LSInfo.h`

LSCopyKindStringForURL

Obtains the kind string for an item designated by URL.

```
OSStatus LSCopyKindStringForURL (
   CFURLRef inURL,
   CFStringRef *outKindString
);
```

Parameters

inURL

A Core Foundation URL reference designating the item whose kind string is requested; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type.

outKindString

A pointer to a Core Foundation string object that, on return, will contain the item's kind string; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. You are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The kind string (which may be localized) is obtained from the item's preferred application, if one is found in the Launch Services database; otherwise, a more generic kind string is chosen. For example, the kind string might be `FrameMaker Document`, or just `Document` if the item is a document for which no application is found. If the item URL's scheme is `file` (designating either a file or a directory), the selection of the preferred application is based on the designated item's filename extension, file type, and creator signature; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`).

Version Notes
Thread-safe since Mac OS version 10.2.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSInfo.h`

LSFindApplicationForInfo

Locates an application with a specified creator signature, bundle ID, filename, or any combination of these characteristics.

```
OSStatus LSFindApplicationForInfo (
   OSType inCreator,
   CFStringRef inBundleID,
   CFStringRef inName,
   FSRef *outAppRef,
   CFURLRef *outAppURL
);
```

Parameters

inCreator

The creator signature to consider. Comparison of creator signatures is case-sensitive. Pass `kLSUnknownCreator` if the application's creator signature is unimportant.

inBundleID

A Core Foundation string object specifying the bundle ID to consider; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. Comparison of bundle IDs is case-insensitive. Pass `NULL` if the application's bundle ID is unimportant.

inName

A Core Foundation string object specifying the filename to consider; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. The string must include any extension (such as `'.app'`) that is part of the filename. Comparison of filenames is case-insensitive. Pass `NULL` if the application's filename is unimportant.

outAppRef

A pointer to a file-system reference that, on return, will identify the requested application; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Pass `NULL` if you are not interested in identifying the application in this form; however, this parameter and `outAppURL` cannot both be `NULL`.

outAppURL

A pointer to a Core Foundation URL reference that, on return, will identify the requested application; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. Pass `NULL` if you are not interested in identifying the application in this form; however, this parameter and `outAppRef` cannot both be `NULL`.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57). If no suitable application is found in the Launch Services database, the function will return the result code `kLSApplicationNotFoundErr`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Discussion
You can request any combination of one, two, or all three of the characteristics specified by the *inCreator*, *inBundleID*, and *inName* parameters; at least one of these characteristics must be supplied. If more than one application is found matching the specified characteristics, Launch Services chooses one in the same manner as when locating the preferred application for opening an item.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSInfo.h`

LSGetApplicationForInfo

Locates the preferred application for opening items with a specified file type, creator signature, filename extension, or any combination of these characteristics.

```
OSStatus LSGetApplicationForInfo (
   OSType inType,
   OSType inCreator,
   CFStringRef inExtension,
   LSRolesMask inRoleMask,
   FSRef *outAppRef,
   CFURLRef *outAppURL
);
```

Parameters

inType

The file type to consider. Comparison of file types is case-sensitive. Pass `kLSUnknownType` if the items' file type is unimportant.

inCreator

The creator signature to consider. Comparison of creator signatures is case-sensitive. Pass `kLSUnknownCreator` if the items' creator signature is unimportant.

inExtension

A Core Foundation string object specifying the filename extension to consider; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type. Comparison of filename extensions is case-insensitive. Pass `NULL` if the items' filename extension is unimportant.

inRolesMask

A bit mask specifying the application's desired role or roles with respect to items with the specified characteristics; see "Roles Mask" (page 46) for a description of this mask. If the role is unimportant, pass `kLSRolesAll`.

outAppRef

A pointer to a file-system reference that, on return, will identify the preferred application for opening items with the specified characteristics; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppURL` cannot both be `NULL`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

outAppURL

A pointer to a Core Foundation URL reference that, on return, will identify the preferred application for items with the specified characteristics; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppRef` cannot both be `NULL`.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value

A result code; see "Launch Services Result Codes" (page 57). If no application suitable for opening items with the specified characteristics is found in the Launch Services database, the function will return the result code `kLSApplicationNotFoundErr`.

Discussion

You can request any combination of one, two, or all three of the characteristics specified by the *inType*, *inCreator*, and *inExtension* parameters; at least one of these characteristics must be supplied. Note that since the choice of a preferred application is subject to any document binding preferences the user may have set, the application chosen will not necessarily be the default application that matches the input characteristics, but may instead be a user-specified application that overrides the default.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`LSInfo.h`

LSGetApplicationForItem

Locates the preferred application for opening an item designated by file-system reference.

```
OSStatus LSGetApplicationForItem (
   const FSRef *inItemRef,
   LSRolesMask inRoleMask,
   FSRef *outAppRef,
   CFURLRef *outAppURL
);
```

Parameters

inItemRef

A pointer to a file-system reference designating the item whose preferred application is requested; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

inRolesMask

A bit mask specifying the application's desired role or roles with respect to the designated item; see "Roles Mask" (page 46) for a description of this mask. If the role is unimportant, pass `kLSRolesAll`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

outAppRef

A pointer to a file-system reference that, on return, will identify the item's preferred application; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppURL` cannot both be `NULL`.

outAppURL

A pointer to a Core Foundation URL reference that, on return, will identify the item's preferred application; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppRef` cannot both be `NULL`.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value

A result code; see "Launch Services Result Codes" (page 57). If no application suitable for opening the item is found in the Launch Services database, the function will return the result code `kLSApplicationNotFoundErr`.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`LSInfo.h`

LSGetApplicationForURL

Locates the preferred application for opening an item designated by URL.

```
OSStatus LSGetApplicationForURL (
   CFURLRef inURL,
   LSRolesMask inRoleMask,
   FSRef *outAppRef,
   CFURLRef *outAppURL
);
```

Parameters

inURL

A Core Foundation URL reference designating the item whose preferred application is requested; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type.

inRolesMask

A bit mask specifying the application's desired role or roles with respect to the designated item; see "Roles Mask" (page 46) for a description of this mask. This parameter applies only to URLs with a scheme component of `file`, and is ignored for all other schemes. If the role is unimportant, pass `kLSRolesAll`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

outAppRef

A pointer to a file-system reference that, on return, will identify the item's preferred application; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppURL` cannot both be `NULL`.

outAppURL

A pointer to a Core Foundation URL reference that, on return, will identify the item's preferred application; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. Pass `NULL` if you are not interested in identifying the preferred application in this form; however, this parameter and `outAppRef` cannot both be `NULL`.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value

A result code; see "Launch Services Result Codes" (page 57). If no application suitable for opening the item is found in the Launch Services database, the function will return the result code `kLSApplicationNotFoundErr`.

Discussion

If the item URL's scheme is `file` (designating either a file or a directory), the selection of the preferred application is based on the designated item's filename extension, file type, and creator signature, along with the role specified by the `inRolesMask` parameter; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`).

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`LSInfo.h`

LSGetExtensionInfo

Obtains the starting index of the extension within a filename.

```
OSStatus LSGetExtensionInfo (
   UniCharCount inNameLen,
   const UniChar inNameBuffer[],
   UniCharCount *outExtStartIndex
);
```

Parameters

inNameLen

The number of characters in the filename specified by the *inNameBuffer* parameter.

inNameBuffer

The buffer containing the filename's Unicode characters.

outExtStartIndex

A pointer to a value of type `UniCharCount` that, on return, will give the starting index of the extension within the filename. If the name does not contain a valid extension (one with no spaces in it), the value on return will be `kLSInvalidExtensionIndex`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The starting index is the number of Unicode characters from the start of the filename buffer to the first character of the extension (not including the period).

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.1 and later.

Related Sample Code
QTCarbonShell

Declared In
`LSInfo.h`

LSGetHandlerOptionsForContentType

Gets the handler options for the specified content type.

```
LSHandlerOptions LSGetHandlerOptionsForContentType (
   CFStringRef inContentType
);
```

Parameters

inContentType
The content type for which the handler options are to be obtained. The content type is a uniform type identifier (UTI).

Return Value
The handler option that is set for the specified content type. For possible values, see "Handler Option Constants" (page 55).

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSOpenApplication

Launches the specified application.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
OSStatus LSOpenApplication (
   const LSApplicationParameters *appParams,
   ProcessSerialNumber *outPSN
);
```

Parameters

inAppParams

A `LSApplicationParameters` (page 42) structure specifying the application to launch and its launch parameters. This parameter cannot be `NULL`.

outPSN

On input, a pointer to a value of type `ProcessSerialNumber` that, on return, contains the process serial number (PSN) of the application specified by `inAppParams`, or `NULL` if you don't want to receive the PSN.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The `LSOpenApplication` launches one application. This function is an updated alternative to the Process Manager's `LaunchApplication` function. Launch arguments are specified in the `inAppParams` argument, which must be supplied. If the application is already running in the current session, it is made the front process (unless the `kLSLaunchNewInstance` flag is used, which always causes a new process to be created).

If `outPSN` is not `NULL`, on return, the structure it points to contains the PSN of the launched (or activated) process. Note that for asynchronous launches, the application may not have finished launching when this function returns.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSOpen.h`

LSOpenCFURLRef

Opens an item designated by URL, in the default manner in its preferred application.

```
OSStatus LSOpenCFURLRef (
   CFURLRef inURL,
   CFURLRef *outLaunchedURL
);
```

Parameters

inURL

A Core Foundation URL reference designating the item to open; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

outLaunchedURL

A pointer to a Core Foundation URL reference that, on return, will identify the application launched. Pass `NULL` if this information is unimportant.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
The designated item is opened in the default manner, as if it had been opened with the `LSOpenFromURLSpec` function with a launch specification specifying the launch flag `kLSLaunchDefaults`: that is, asynchronously, starting the Classic emulation environment if necessary, and with the remaining launch parameters taken from the application's information property list. For greater control, call `LSOpenFromURLSpec` directly. See "Launch Flags" (page 47) for more information about launch flags.

If the item URL's scheme is `file` (designating either a file or a directory), the selection of the preferred application is based on the designated item's filename extension, file type, and creator signature; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`). The application is launched or activated, as required, and sent an appropriate Apple event depending on the circumstances:

- If the URL's scheme is `file` and it designates a document, the document's preferred application is launched (or activated if it is already running).
  - If the application claims to accept `file` URLs, it is sent a `'GURL'` ("get URL") Apple event containing the item's URL.
  - If the application does not claim to accept `file` URLs, it is sent an `'odoc'` ("open document") Apple event identifying the document to open.
- If the URL's scheme is `file` and it designates 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 and sent an `'rapp'` ("reopen application") Apple event.
- If the URL has a scheme other than `file`, the scheme's preferred application is launched (or activated if it is already running) and sent a `'GURL'` ("get URL") Apple event containing the item's URL.

As of Mac OS X v10.4 and later, `LSOpenURLsWithRole` (page 36) is the preferred way of opening a URL.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
BSDLLCTest
QTCarbonShell
SimpleDataQueue
SimpleUserClient

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Declared In
`LSOpen.h`

LSOpenFromRefSpec

Opens one or more items designated by file-system reference, in either their preferred applications or a designated application.

```
OSStatus LSOpenFromRefSpec (
   const LSLaunchFSRefSpec *inLaunchSpec,
   FSRef *outLaunchedRef
);
```

Parameters

inLaunchSpec

A pointer to a file-based launch specification indicating what to open and how to launch the relevant application or applications; see LSLaunchFSRefSpec (page 43) for a description of this structure.

outLaunchedRef

A pointer to a file-system reference that, on return, will identify the application launched; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Pass `NULL` if this information is unimportant. If more than one application is launched, the one identified will be the one corresponding to the first item designated in the launch specification.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
This function affords greater control of how items are opened or applications launched than is possible with the `LSOpenFSRef` function. For instance, you can use it to open multiple items in a single call, in either the same or different applications; open documents for printing rather than for simple viewing or editing; or force a document to open in an application other than its own preferred application.

The launch specification supplied for the `inLaunchSpec` parameter may designate an application to launch, items to open, or both. The relevant application or applications are launched or activated, as required, and sent an appropriate Apple event depending on the circumstances:

- If the launch specification designates both items to open and an application with which to open them, the designated application is used to open all of the items. The application is launched (or activated if it is already running) and sent an `'odoc'` ("open document") Apple event containing the list of items to open; if the items are to be printed, the Apple event is `'pdoc'` ("print document") instead.

  Note: When both an application and a list of items are supplied, the designated application is asked to open all of the items, whether or not it claims the ability to do so. Launch Services does not report an error if the application is unable to open one or more of the items; any error processing is the application's responsibility.

- If the launch specification designates items to open but not an application with which to open them, each item is opened in its own preferred application. Each application is launched or activated and sent an `'odoc'` or `'pdoc'` Apple event, as described for the preceding case. (If two or more of the items have the same preferred application, the application receives a single `'odoc'` or `'pdoc'` event listing all of the relevant items.)

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

- If the launch specification designates only an application to launch (or if one or more of the items to open are applications):
  - 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 and sent an `'rapp'` ("reopen application") Apple event.

As of Mac OS X v10.4 and later, `LSOpenItemsWithRole` (page 34) is the preferred way of opening items.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSOpen.h`

LSOpenFromURLSpec

Opens one or more items designated by URL, in either their preferred applications or a designated application.

```
OSStatus LSOpenFromURLSpec (
   const LSLaunchURLSpec *inLaunchSpec,
   CFURLRef *outLaunchedURL
);
```

Parameters

inLaunchSpec

A pointer to a URL-based launch specification indicating what to open and how to launch the relevant application or applications; see LSLaunchURLSpec (page 44) for a description of this structure.

outLaunchedURL

A pointer to a Core Foundation URL reference that, on return, will identify the application launched; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. Pass `NULL` if this information is unimportant. If more than one application is launched, the one identified will be the one corresponding to the first item designated in the launch specification.

Despite the absence of the word `Copy` in its name, this function retains the URL reference object on your behalf; you are responsible for releasing this object.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
This function affords greater control of how items are opened or applications launched than is possible with the `LSOpenCFURLRef` function. For instance, you can use it to open multiple items in a single call, in either the same or different applications; open documents for printing rather than for simple viewing or editing; or force a document to open in an application other than its own preferred application.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

The launch specification supplied for the `inLaunchSpec` parameter may designate an application to launch, items to open, or both. The relevant application or applications are launched or activated, as required, and sent an appropriate Apple event depending on the circumstances:

- If the launch specification designates both items to open and an application with which to open them, the designated application is used to open all of the items. The application is launched (or activated if it is already running) and sent one or more Apple events:
  - If one or more of the item URLs have scheme `file` and designate documents to open, and if the application claims to accept `file` URLs, it is sent a `'GURL'` ("get URL") Apple event for each such URL.
  - If one or more of the item URLs have scheme `file` and designate documents to open, and if the application does not claim to accept `file` URLs, it is sent a single `'odoc'` ("open document") Apple event containing the list of items to open; if the items are to be printed, the Apple event is `'pdoc'` ("print document") instead.
  - For each item URL with a scheme other than `file`, the application is sent a `'GURL'` ("get URL") Apple event containing the item's URL.

> **Note:** When both an application and a list of items are supplied, the designated application is asked to open all of the items, whether or not it claims the ability to do so. Launch Services does not report an error if the application is unable to open one or more of the items; any error processing is the application's responsibility.

- If the launch specification designates items to open but not an application with which to open them, each item is opened in its own preferred application. Each application is launched or activated and sent one or more Apple events, as described for the preceding case. (If two or more of the items have the same preferred application, the application receives a single `'odoc'` or `'pdoc'` event listing all of the relevant items.)
- If the launch specification designates only an application to launch (or if one or more of the items to open are `file` URLs designating applications):
  - 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 and sent an `'rapp'` ("reopen application") Apple event.

As of Mac OS X v10.4 and later, `LSOpenURLsWithRole` (page 36) is the preferred way of opening URLs.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSOpen.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSOpenFSRef

Opens an item designated by file-system reference, in the default manner in its preferred application.

```
OSStatus LSOpenFSRef (
   const FSRef *inRef,
   FSRef *outLaunchedRef
);
```

Parameters

inRef

A pointer to a file-system reference designating the item to open; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

outLaunchedRef

A pointer to a file-system reference that, on return, will identify the application launched. Pass `NULL` if this information is unimportant.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Discussion

The designated item is opened in the default manner, as if it had been opened with the `LSOpenFromRefSpec` function with a launch specification specifying the launch flag `kLSLaunchDefaults`: that is, asynchronously, starting the Classic emulation environment if necessary, and with the remaining launch parameters taken from the application's information property list. For greater control, call `LSOpenFromRefSpec` directly. See "`Launch Flags`" (page 47) for more information about launch flags.

The application is launched or activated, as required, and sent an appropriate Apple event depending on the circumstances:

- If the item is a document, its preferred application is launched (or activated if it is already running) and sent an `'odoc'` ("open document") Apple event.
- If the item is an application that is not already running, it is launched and sent an `'oapp'` ("open application") Apple event.
- If the item is an application that is already running, it is activated and sent an `'rapp'` ("reopen application") Apple event.

As of Mac OS X v10.4 and later, `LSOpenItemsWithRole` (page 34) is the preferred way of opening an item.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`LSOpen.h`

LSOpenItemsWithRole

Opens items specified as an array of values of type `FSRef` with a specified role.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
OSStatus LSOpenItemsWithRole (
   const FSRef *inItems,
   CFIndex inItemCount,
   LSRolesMask inRole,
   const AEKeyDesc *inAEParam,
   const LSApplicationParameters *inAppParams,
   ProcessSerialNumber *outPSNs,
   CFIndex inMaxPSNCount
);
```

Parameters

inItems

An array of values of type `FSRef`.

inItemCount

The number of items specified in `inItems`.

inRole

A value of type `LSRolesMask` specifying one or more roles. If the role doesn't matter, use `kLSRolesAll`. For possible values, see "Roles Mask" (page 46). If the `inAppParams` parameter is not `NULL`, this parameter is ignored.

inAEParam

An `AEKeyDesc` that is to be attached to the Apple Event(s) generated by Launch Services with the specified `AEKeyword`. This parameter can be `NULL`.

inAppParams

An `LSApplicationParameters` (page 42) structure specifying the application to launch and its launch parameters, in which case the `inRole` parameter is ignored. This parameter can be `NULL`, in which case an application is selected that can handle each input item in at least one of the roles specified by the `inRole` parameter.

outPSNs

On input, a pointer to a caller-allocated buffer or `NULL` if you don't want to receive process serial number (PSN) information. If not `NULL` on input, on return, the buffer contains at each index the PSN that was used to open the item at the same index of the input item array (`inItems`).

inMaxPSNCount

The maximum number of PSNs that the buffer pointed to by `outPSNs` can hold.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Discussion

This function opens the specified items with the specified role. You can optionally specify the application and launch parameters in the `inAppParams` parameter. If an application is specified in the `inAppParams` parameter, the `inRole` parameter is ignored and the application is launched (if necessary).

Each application (regardless of whether it is launched or already running) receives an `'odoc'` Apple Event specifying the items that are to be opened.

If the `inItems` array contains any applications, this function launches them only if the `kLSRolesShell` bit is set in the `inRoles` parameter to indicate that the operating system should use the application itself as the execution shell of the new process.

If not `NULL`, the `outPSNs` buffer is filled with the PSN that was used to open each item at the same index of the `inItems` array. The PSN capacity of the output buffer is specified by `inMaxPSNCount`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSOpen.h`

LSOpenURLsWithRole

Opens one or more URLs with the specified roles.

```
OSStatus LSOpenURLsWithRole (
   CFArrayRef inURLs,
   LSRolesMask inRole,
   const AEKeyDesc *inAEParam,
   const LSApplicationParameters *inAppParams,
   ProcessSerialNumber *outPSNs,
   CFIndex inMaxPSNCount
);
```

Parameters

inURLs

An array of values of type `CFURLRef`.

inRole

A value of type `LSRolesMask` specifying one or more roles. If the role doesn't matter, use `kLSRolesAll`. For possible values, see "Roles Mask" (page 46). This parameter is ignored if the `inAppParams` parameter is not `NULL`.

inAEParam

A value of type `AEKeyDesc` that is to be attached to the Apple Event(s) generated by Launch Services with the specified `AEKeyword`. This parameter can be `NULL`.

inAppParams

An `LSApplicationParameters` (page 42) structure specifying the application to launch and its launch parameters, in which case the `inRole` parameter is ignored. This parameter can be `NULL`, in which case an application is selected that can handle each input URL in at least one of the roles specified by the `inRole` parameter.

outPSNs

On input, a pointer to a caller-allocated buffer or `NULL` if you don't want to receive process serial number (PSN) information. If not `NULL` on input, on return, the buffer contains at each index the PSN that was used to open the URL at the same index of the input URL array (`inURLs`).

inMaxPSNCount

The maximum number of PSNs that the buffer pointed to by `outPSNs` can hold.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
This function opens the URLs specified by `inURLs` with the roles specified by `inRole`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Each launched application receives one or more 'GURL' Apple Events specifying the URLs to be opened. You can also pass file URLs, which are interpreted as file system items and opened in the manner of LSOpenItemsWithRole (page 34) (that is, a handler is selected based on the item's filesystem metadata).If inURLs contains any application URLs, this function launches them only if the kLSRolesShell bit is set in the inRoles parameter, in which case the application is its own shell.If not NULL, the outPSNs buffer is filled with the process serial numbers that were used to open each URL at the same index of the input URL array specified by the inURLs parameter. The PSN capacity of the output buffer is specified by inMaxPSNCount.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
LSOpen.h

LSRegisterFSRef

Registers an application, designated by file-system reference, in the Launch Services database.

```
OSStatus LSRegisterFSRef (
   const FSRef *inRef,
   Boolean inUpdate
);
```

Parameters

inRef

A pointer to a file-system reference designating the application to be registered; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the FSRef data type.

inUpdate

A Boolean value specifying whether Launch Services should update existing information registered for the application, if any. If this parameter is false, the application will not be registered if it has already been registered previously and its current modification date has not changed from when it was last registered; if the parameter is true, the application's registered information will be updated even if its modification date has not changed.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
This function adds the designated application and its document and URL claims (if any) to the Launch Services database, making the application a candidate for document and URL binding.

Version Notes
Thread-safe since Mac OS version 10.3.

Availability
Available in Mac OS X v10.3 and later.

Declared In
LSInfo.h

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSRegisterURL

Registers an application, designated by URL, in the Launch Services database.

```
OSStatus LSRegisterURL (
   CFURLRef inURL,
   Boolean inUpdate
);
```

Parameters

inFileURL

A Core Foundation URL reference designating the application to be registered; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to an application file or application bundle.

inUpdate

A Boolean value specifying whether Launch Services should update existing information registered for the application, if any. If this parameter is `false`, the application will not be registered if it has already been registered previously and its current modification date has not changed from when it was last registered; if the parameter is `true`, the application's registered information will be updated even if its modification date has not changed.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

Discussion

This function adds the designated application and its document and URL claims (if any) to the Launch Services database, making the application a candidate for document and URL binding.

Version Notes

Thread-safe since Mac OS version 10.3.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`LSInfo.h`

LSSetDefaultHandlerForURLScheme

Sets the user's preferred default handler for the specified URL scheme.

```
OSStatus LSSetDefaultHandlerForURLScheme (
   CFStringRef inURLScheme,
   CFStringRef inHandlerBundleID
);
```

Parameters

inURLScheme

The URL scheme for which the handler is to be set.

inHandlerBundleID

The application bundle identifier that is to be set as the handler for the URL scheme specified by `inURLScheme`.

Return Value

A result code; see "Launch Services Result Codes" (page 57).

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Discussion
Call `LSCopyDefaultHandlerForURLScheme` (page 13) to get the current setting of the user's preferred default handler for a specified content type.

URL handling capability is determined according to the value of the `CFBundleURLTypes` key in an application's `Info.plist`. For information on the `CFBundleURLTypes` key, see the section "CFBundleURLTypes" in *Mac OS X Runtime Configuration Guidelines*.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

LSSetDefaultRoleHandlerForContentType

Sets the user's preferred default handler for the specified content type in the specified roles.

```
OSStatus LSSetDefaultRoleHandlerForContentType (
   CFStringRef inContentType,
   LSRolesMask inRole,
   CFStringRef inHandlerBundleID
);
```

Parameters

inContentType
The content type for which the default role handler is being set. The content type is a uniform type identifier (UTI).

inRole
The roles for which the default role handler is being set. Pass `kLSRolesAll` to specify all roles. For additional possible values, see "Roles Mask" (page 46).

inHandlerBundleID
The application bundle identifier that is to be set as the default handler for the specified content type and roles.

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Discussion
Call `LSCopyDefaultRoleHandlerForContentType` (page 14) to get the current setting of the user's preferred default handler for a specified content type.

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

LSSetExtensionHiddenForRef

Specifies whether the filename extension for an item designated by file-system reference should be hidden or shown.

```
OSStatus LSSetExtensionHiddenForRef (
   const FSRef *inRef,
   Boolean inHide
);
```

Parameters

inRef

A pointer to a file-system reference designating the item whose filename extension is to be hidden or shown; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type.

inHide

A Boolean value specifying whether the filename extension should be hidden (`true`) or shown (`false`).

Return Value

A result code; see "Launch Services Result Codes" (page 57). The function will return the result code `kLSCannotSetInfoErr` if:

- The extension is not valid (contains spaces)
- The extension is not active (is not claimed by an application registered with Launch Services)
- Hiding the extension would make the filename appear to have an active but incorrect extension (for example, in the filename `Photo.jpeg.scpt`, where hiding the extension would make an AppleScript file appear to be a JPEG file)

Discussion

This function sets the necessary file-system state controlling whether the filename extension should be hidden in the display name of the item designated by the `inRef` parameter. To determine whether an item's extension is currently hidden, you can use the `LSCopyItemInfoForRef` function.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`LSInfo.h`

LSSetExtensionHiddenForURL

Specifies whether the filename extension for an item designated by URL should be hidden or shown.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
OSStatus LSSetExtensionHiddenForURL (
   CFURLRef inURL,
   Boolean inHide
);
```

Parameters

inFileURL

A Core Foundation URL reference designating the item whose filename extension is to be hidden or shown; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to either a file or a directory.

inHide

A Boolean value specifying whether the extension should be hidden (`true`) or shown (`false`).

Return Value

A result code; see "Launch Services Result Codes" (page 57). The function will return the result code `kLSCannotSetInfoErr` if:

- The extension is not valid (contains spaces)
- The extension is not active (is not claimed by an application registered with Launch Services)
- Hiding the extension would make the filename appear to have an active but incorrect extension (for example, in the filename `Photo.jpeg.scpt`, where hiding the extension would make an AppleScript file appear to be a JPEG file)

Discussion

This function sets the necessary file-system state controlling whether the filename extension should be hidden in the display name of the item designated by the `inFileURL` parameter. To determine whether an item's extension is currently hidden, you can use the `LSCopyItemInfoForURL` function.

Version Notes

Thread-safe since Mac OS version 10.2.

Availability

Available in Mac OS X v10.1 and later.

Declared In

`LSInfo.h`

LSSetHandlerOptionsForContentType

Sets the handler option for the specified content type.

```
OSStatus LSSetHandlerOptionsForContentType (
   CFStringRef inContentType,
   LSHandlerOptions inOptions
);
```

Parameters

inContentType

The content type for which the handler options are to be set. The content type is a uniform type identifier (UTI).

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

inOptions
The handler options to set. For possible values, see "Handler Option Constants" (page 55).

Return Value
A result code; see "Launch Services Result Codes" (page 57).

Version Notes
Thread-safe since Mac OS X v10.4.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSInfo.h`

Data Types

This section describes the data types defined in the Launch Services API.

LSApplicationParameters

Specifies the application, launch flags, and additional parameters that control how an application is launched.

```
struct LSApplicationParameters {
    CFIndex version;
    LSLaunchFlags flags;
    const FSRef * application;
    void * asyncLaunchRefCon;
    CFDictionaryRef environment;
    CFArrayRef argv;
    AppleEvent * initialEvent
};
typedef struct LSApplicationParameters LSApplicationParameters;
```

Fields
`version`
The version of the structure. The value of this field must be `0`.

`flags`
Launch flags. For possible values, see "Launch Flags" (page 47).

`application`
The `FSRef` of the application to open.

`asyncLaunchRefCon`
The client `refCon` that is to appear in subsequent launch notifications.

`environment`
A dictionary of `CFStringRef` keys and values for environment variables to set in the launched process. The value of this field can be `NULL`.

`argv`
An array of values of type `CFStringRef` that specify the arguments that are to be passed to `main()` in the launched process. The value of this field can be `NULL`. This field is ignored in Mac OS X v10.4.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`initialEvent`
: The first Apple Event to send to the launched process. The value of this field can be `NULL`.

Discussion
This structure is passed as a parameter to `LSOpenApplication` (page 28), `LSOpenItemsWithRole` (page 34), and `LSOpenURLsWithRole` (page 36).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`LSOpen.h`

LSLaunchFSRefSpec

Specifies, by file-system reference, an application to launch, items to open, or both, along with related information.

```
struct LSLaunchFSRefSpec {
    const FSRef *appRef;
    UInt32 numDocs;
    const FSRef *itemRefs;
    const AEDesc *passThruParams;
    LSLaunchFlags launchFlags;
    void *asyncRefCon;
};
typedef struct LSLaunchFSRefSpec LSLaunchFSRefSpec;
```

Fields

`appRef`
: A pointer to a file-system reference designating the application to launch; see the *File Manager Reference* in the Carbon File Management Documentation for a description of the `FSRef` data type. Set this field to `NULL` to request that each item in the `itemRefs` array be opened in its own preferred application.

`numDocs`
: The number of elements in the array specified by the `itemRefs` field. The value of this field can be `0`, in which case the application designated by `appRef` is launched without opening any items.

`itemRefs`
: An array of file-system references designating the item or items to open. If the value of `numDocs` is `0`, this field is ignored and can be set to `NULL`.

`passThruParams`
: A pointer to an Apple event descriptor that is passed untouched as an optional parameter, with keyword `keyAEPropData` (`'prdt'`), in the Apple event sent to each application launched or activated (whether individual preferred applications or the application designated by `appRef`). See the *Apple Event Manager Reference* in the Carbon Interapplication Communication Documentation for a description of the `AEDesc` data type. The value of this field can be `NULL`.

`launchFlags`
: Launch flags specifying how to launch each application (including whether to print or merely open documents); see "Launch Flags" (page 47) for a description of these flags.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`asyncRefCon`

A pointer to an arbitrary application-defined value, passed in the Carbon event notifying you of an application's launch or termination (if you have registered for such notification). The value of this field can be `NULL`.

Discussion

This data type defines a file-based launch specification designating, by file-system reference, an application to launch, items to open, or both. To request that items be opened in a particular application, set `appRef`, `numDocs`, and `itemRefs` accordingly. To request that each designated item be opened in its own preferred application, set *`appRef`* to `NULL`. To request that a particular application be launched without opening any documents, set *`appRef`* accordingly and set `numDocs` to `0`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`LSOpen.h`

LSLaunchURLSpec

Specifies, by URL, an application to launch, items to open, or both, along with related information.

```
struct LSLaunchURLSpec {
    CFURLRef appURL;
    CFArrayRef itemURLs;
    const AEDesc *passThruParams;
    LSLaunchFlags launchFlags;
    void *asyncRefCon;
};
typedef struct LSLaunchURLSpec LSLaunchURLSpec;
```

Fields

`appURL`

A Core Foundation URL reference designating the application to launch; see the *CFURL Reference* in the Core Foundation Reference Documentation for a description of the `CFURLRef` data type. The URL must have scheme `file` and contain a valid path to an application file or application bundle. Set this field to `NULL` to request that each item in the `itemURLs` array be opened in its own preferred application.

`itemURLs`

A reference to an array of Core Foundation URL references designating the item or items to open; see the *CFArray Reference* in the Core Foundation Reference Documentation for a description of the `CFArrayRef` data type. The value of this field can be `NULL`, in which case the application designated by `appURL` will be launched without opening any items.

`passThruParams`

A pointer to an Apple event descriptor that is passed untouched as an optional parameter, with keyword `keyAEPropData` (`'prdt'`), in the Apple event sent to each application launched or activated (whether individual preferred applications or the application designated by `appURL`). See the *Apple Event Manager Reference* in the Carbon Interapplication Communication Documentation for a description of the `AEDesc` data type. The value of this field can be `NULL`.

`launchFlags`

Launch flags specifying how to launch each application (including whether to print or merely open documents); see "Launch Flags" (page 47) for a description of these flags.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`asyncRefCon`

A pointer to an arbitrary application-defined value, passed in the Carbon event notifying you of an application's launch or termination (if you have registered for such notification). The value of this field can be `NULL`.

Discussion
This data type defines a URL-based launch specification designating, by URL, an application to launch, items to open, or both. To request that items be opened in a particular application, set *appURL* and `itemURLs` accordingly. To request that each designated item be opened in its own preferred application, set *appURL* to `NULL`. If the item URL's scheme is `file` (designating either a file or a directory), the selection of the preferred application is based on the designated item's filename extension, file type, and creator signature; otherwise, it is based on the URL scheme (such as `http`, `ftp`, or `mailto`). To request that a particular application be launched without opening any document, set *appURL* accordingly and set `itemURLs` to `NULL`.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSOpen.h`

LSItemInfoRecord

Contains requested information about an item.

```
struct LSItemInfoRecord {
    LSItemInfoFlags flags;
    OSType filetype;
    OSType creator;
    CFStringRef extension;
    CFStringRef iconFileName;
    LSKindID kindID;
};
typedef struct LSItemInfoRecord LSItemInfoRecord;
```

Fields
`flags`

Item-information flags specifying information about the item; see "Item-Information Flags" (page 52) for a description of these flags.

`filetype`

The item's file type.

`creator`

The item's creator signature.

`extension`

A Core Foundation string object specifying the item's filename extension; see the *CFString Reference* in the Core Foundation Reference Documentation for a description of the `CFStringRef` data type.

`iconFileName`

No longer used.

`kindID`

No longer used.

Discussion
This data type defines an item-information record used by the `LSCopyItemInfoForRef` and `LSCopyItemInfoForURL` functions to return requested information about an item.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`LSInfo.h`

LSKindID

Data type of the `kindID` field of an item-information record (`LSItemInfoRecord`); no longer used.

```
typedef UInt32 LSKindID;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LSInfo.h`

Constants

This section describes the constants defined in the Launch Services API.

Roles Mask

Specify the desired role or roles for an application to claim with respect to an item or family of items.

```
typedef OptionBits LSRolesMask;enum {
    kLSRolesNone = 0x00000001,
    kLSRolesViewer = 0x00000002,
    kLSRolesEditor = 0x00000004,
    kLSRolesShell = 0x00000008,
    kLSRolesAll = 0xFFFFFFFF
};
```

Constants

`kLSRolesNone`

Requests the role `None` (the application cannot open the item, but provides an icon and a kind string for it).

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRolesViewer`

Requests the role `Viewer` (the application can read and present the item, but cannot manipulate or save it).

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSRolesEditor`
: Requests the role `Editor` (the application can read, present, manipulate, and save the item).

  Available in Mac OS X v10.0 and later.

  Declared in `LSInfo.h`.

`kLSRolesShell`
: Requests the role `Shell` (the application can execute the item).

  Available in Mac OS X v10.4 and later.

  Declared in `LSInfo.h`.

`kLSRolesAll`
: Accepts any role with respect to the item.

  Available in Mac OS X v10.0 and later.

  Declared in `LSInfo.h`.

Discussion
This bit mask is passed to functions that find the preferred application for a given item or family of items (`LSGetApplicationForItem`, `LSGetApplicationForURL`, `LSGetApplicationForInfo`), or that determine whether a given application can open a designated item (`LSCanRefAcceptItem`, `LSCanURLAcceptURL`), to specify the application's desired role or roles with respect to the item. For example, to request only an editor application, specify `kLSRolesEditor`; if either an editor or a viewer application is acceptable, specify `kLSRolesEditor | kLSRolesViewer`.

Launch Flags

Specify how to launch an application.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
typedef OptionBits LSLaunchFlags;enum {
    kLSLaunchDefaults = 0x00000001,
    kLSLaunchAndPrint = 0x00000002,
    kLSLaunchReserved2 = 0x00000004,
    kLSLaunchReserved3 = 0x00000008,
    kLSLaunchReserved4 = 0x00000010,
    kLSLaunchReserved5 = 0x00000020,
    kLSLaunchAndDisplayErrors = 0x00000040,
    kLSLaunchInhibitBGOnly = 0x00000080,
    kLSLaunchDontAddToRecents = 0x00000100,
    kLSLaunchDontSwitch = 0x00000200,
    kLSLaunchNoParams = 0x00000800,
    kLSLaunchAsync = 0x00010000,
    kLSLaunchStartClassic = 0x00020000,
    kLSLaunchInClassic = 0x00040000,
    kLSLaunchNewInstance = 0x00080000,
    kLSLaunchAndHide = 0x00100000,
    kLSLaunchAndHideOthers = 0x00200000,
    kLSLaunchHasUntrustedContents = 0x00400000
};
```

Constants

`kLSLaunchDefaults`

Requests launching in the default manner (as if the only flags set were `kLSLaunchNoParams`, `kLSLaunchAsync`, and `kLSLaunchStartClassic`).

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchAndPrint`

Requests that documents opened in the application be printed.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchReserved2`

Reserved for future use.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchReserved3`

Reserved for future use.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchReserved4`

Reserved for future use.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchReserved5`

Reserved for future use.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSLaunchAndDisplayErrors`
Requests that launch and open failures be displayed in the UI.

Available in Mac OS X v10.4 and later.

Declared in `LSOpen.h`.

`kLSLaunchInhibitBGOnly`
Requests that the launch be made to fail if the application is background-only.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchDontAddToRecents`
Requests that the application or documents not be added to the Finder's Recent Items menu.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchDontSwitch`
Requests that the application be launched without being brought to the foreground.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchNoParams`
Requests that the application's information property list be used to determine the launch parameters.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchAsync`
Requests that the application be launched asynchronously: that is, the Launch Services function launching it return control immediately, without waiting for it to complete its launch sequence (indicated visually to the user when the application's icon stops "bouncing" in the Dock).

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchStartClassic`
Requests that the Classic emulation environment be started up if the application requires it. If this flag is not set and the application requires the Classic environment, the launch will fail.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchInClassic`
Requests that the application be forced to launch in the Classic emulation environment.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchNewInstance`
Requests that a new instance of the application be started, even if one is already running.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchAndHide`
Requests that the application be hidden as soon as it completes its launch sequence.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSLaunchAndHideOthers`
Requests that other applications be hidden as soon as the opened application completes its launch sequence.

Available in Mac OS X v10.0 and later.

Declared in `LSOpen.h`.

`kLSLaunchHasUntrustedContents`
Requests that the items to be launched should be marked as untrusted.

Available in Mac OS X v10.4 and later.

Declared in `LSOpen.h`.

Discussion
They are passed in a launch specification structure (`LSLaunchFSRefSpec` to the `LSOpenFromRefSpec` function or `LSLaunchURLSpec` to the `LSOpenFromURLSpec` function), to control the manner in which applications are launched.

Requested-Information Flags

Specify what information to obtain about an item.

```
typedef OptionBits LSRequestedInfo;enum {
    kLSRequestExtension = 0x00000001,
    kLSRequestTypeCreator = 0x00000002,
    kLSRequestBasicFlagsOnly = 0x00000004,
    kLSRequestAppTypeFlags = 0x00000008,
    kLSRequestAllFlags = 0x00000010,
    kLSRequestIconAndKind = 0x00000020,
    kLSRequestExtensionFlagsOnly = 0x00000040,
    kLSRequestAllInfo = 0xFFFFFFFF
};
```

Constants
`kLSRequestExtension`
Requests the item's filename extension.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRequestTypeCreator`
Requests the item's file type and creator signature.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRequestBasicFlagsOnly`
Requests all item-information flags that are not application-specific: that is, all except `kLSItemInfoIsNativeApp` through `kLSItemInfoAppIsScriptable`.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRequestAppTypeFlags`
Requests all application-specific item-information flags: that is, `kLSItemInfoIsNativeApp` through `kLSItemInfoAppIsScriptable`.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSRequestAllFlags`
Requests all item-information flags.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRequestIconAndKind`
Not used.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSRequestExtensionFlagsOnly`
Requests only the `kLSItemInfoExtensionIsHidden` item-information flag.

Available in Mac OS X v10.1 and later.

Declared in `LSInfo.h`.

`kLSRequestAllInfo`
Requests all available item information.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

Discussion
These flags are passed to the `LSCopyItemInfoForRef` and `LSCopyItemInfoForURL` functions to specify the type of information to be obtained in an item-information record; see LSItemInfoRecord (page 45) for a description of this structure.

Item Attribute Constants

Constants used to retrieve item attributes.

```
const CFStringRef kLSItemContentType;
const CFStringRef kLSItemFileType;
const CFStringRef kLSItemFileCreator;
const CFStringRef kLSItemExtension;
const CFStringRef kLSItemDisplayName
const CFStringRef kLSItemDisplayKind;
const CFStringRef kLSItemRoleHandlerDisplayName;
const CFStringRef kLSItemIsInvisible;
const CFStringRef kLSItemExtensionIsHidden;
```

Constants
`kLSItemContentType`
The item's content type identifier, which is a uniform type identifier string. The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemFileType`
The item's file type (OSType). The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSItemFileCreator`
The item's file creator (OSType). The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemExtension`
The item's filename extension. The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemDisplayName`
The item's name as displayed to the user. The display name reflects localization and extension hiding that may be in effect. The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemDisplayKind`
The localized kind string describing the item's type. The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemRoleHandlerDisplayName`
The display name of the application that is set to handle this item, subject to the role mask. The value type of this attribute is `CFStringRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemIsInvisible`
A value of `kCFBooleanTrue` if the item is normally hidden from users; otherwise, `kCFBooleanFalse`. The value type of this attribute is `CFBooleanRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSItemExtensionIsHidden`
A value of `kCFBooleanTrue` if the item's extension is set to be hidden; otherwise, `kCFBooleanFalse`. The value type of this attribute is `CFBooleanRef`.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

Item-Information Flags

Provide information about an item.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
typedef OptionBits LSItemInfoFlags;enum {
    kLSItemInfoIsPlainFile = 0x00000001,
    kLSItemInfoIsPackage = 0x00000002,
    kLSItemInfoIsApplication = 0x00000004,
    kLSItemInfoIsContainer = 0x00000008,
    kLSItemInfoIsAliasFile = 0x00000010,
    kLSItemInfoIsSymlink = 0x00000020,
    kLSItemInfoIsInvisible = 0x00000040,
    kLSItemInfoIsNativeApp = 0x00000080,
    kLSItemInfoIsClassicApp = 0x00000100,
    kLSItemInfoAppPrefersNative = 0x00000200,
    kLSItemInfoAppPrefersClassic = 0x00000400,
    kLSItemInfoAppIsScriptable = 0x00000800,
    kLSItemInfoIsVolume = 0x00001000,
    kLSItemInfoExtensionIsHidden = 0x00100000
};
```

Constants

`kLSItemInfoIsPlainFile`
Item is a data file (and not, for example, a directory, volume, or UNIX symbolic link).
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsPackage`
Item is a packaged directory.
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsApplication`
Item is a single-file or packaged application.
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsContainer`
Item is a directory (includes packages) or volume.
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsAliasFile`
Item is an alias file (includes symbolic links).
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsSymlink`
Item is a UNIX symbolic link.
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

`kLSItemInfoIsInvisible`
Item is invisible, because either its name begins with a period or its `isInvisible` Finder flag is set.
Available in Mac OS X v10.0 and later.
Declared in `LSInfo.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSItemInfoIsNativeApp`
Item is an application that can run natively in Mac OS X.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoIsClassicApp`
Item is an application that cannot run natively and must be launched in the Classic emulation environment.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoAppPrefersNative`
Item is an application that can run either natively or in the Classic emulation environment, but prefers to be launched natively. This flag is valid only when `kLSItemInfoIsNativeApp` is set.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoAppPrefersClassic`
Item is an application that can run either natively or in the Classic emulation environment, but prefers to be launched in the Classic environment. This flag is valid only when `kLSItemInfoIsNativeApp` is set.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoAppIsScriptable`
Item is an application that can be scripted.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoIsVolume`
Item is the root directory of a volume or mount point.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSItemInfoExtensionIsHidden`
Item has a hidden filename extension.

Available in Mac OS X v10.1 and later.

Declared in `LSInfo.h`.

Discussion
These flags are set in an item-information record to provide information about an item; see LSItemInfoRecord (page 45) for a description of this structure.

Acceptance Flags

Specify behavior to observe when testing whether an application can accept (open) an item.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
typedef OptionBits LSAcceptanceFlags;enum {
    kLSAcceptDefault = 0x00000001,
    kLSAcceptAllowLoginUI = 0x00000002
};
```

Constants

`kLSAcceptDefault`

Requests the default behavior (as opposed to the behavior specified by the `kLSAcceptAllowLoginUI` flag).

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSAcceptAllowLoginUI`

Requests that the user interface to log in be presented if necessary. If `LSCanRefAcceptItem` or `LSCanURLAcceptURL` is called during a drag-and-drop operation, showing a server login dialog would be an inappropriate user experience. If the target designated in the function call is an alias to an application, Launch Services needs to resolve the alias to ascertain what file types the application can open; however, if the application is on a server that needs to be authenticated, Launch Services will by default fail to resolve the alias, to avoid having to present the login interface. To override this default behavior by allowing the server login interface, set the `kLSAcceptAllowLoginUI` flag.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

Discussion

These flags are passed to the functions `LSCanRefAcceptItem` and `LSCanURLAcceptURL`.

Handler Option Constants

Specify the options for controlling how content handlers are selected.

```
typedef OptionBits LSHandlerOptions;enum {
    kLSHandlerOptionsDefault = 0,
    kLSHandlerOptionsIgnoreCreator = 1
};
```

Constants

`kLSHandlerOptionsDefault`

When set, causes Launch Services to use a content item's creator (when available) to select a handler. This is the default setting.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

`kLSHandlerOptionsIgnoreCreator`

When set, causes Launch Services to ignore the content item's creator when selecting a role handler for the specified content type.

Available in Mac OS X v10.4 and later.

Declared in `LSInfo.h`.

Invalid Extension Index

Represents an invalid filename extension index.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

```
enum {
    kLSInvalidExtensionIndex = 0xFFFFFFFF
};
```

Constants

`kLSInvalidExtensionIndex`

The value obtained by the `LSGetExtensionInfo` function if the filename does not contain a valid extension.

Available in Mac OS X v10.1 through Mac OS X v10.4.

Declared in `LSInfo.h`.

Unknown Type or Creator

Represent an unknown file type or creator.

```
enum {
    kLSUnknownType = 0,
    kLSUnknownCreator = 0
};
```

Constants

`kLSUnknownType`

The value to supply as the file type (for example, to the `LSGetApplicationForInfo` function) if no file type information is available.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

`kLSUnknownCreator`

The value to supply as the creator signature if no file creator information is available.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

Constants No Longer Used

The following constants are no longer used.

```
typedef OptionBits LSInitializeFlags;enum {
    kLSInitializeDefaults = 0x00000001
};enum {
    kLSUnknownKindID = 0
};enum {
    kLSMinCatInfoBitmap = (kFSCatInfoNodeFlags | kFSCatInfoParentDirID
| kFSCatInfoFinderInfo | kFSCatInfoFinderXInfo)
};
```

Constants

`kLSInitializeDefaults`

Formerly passed to the `LSInit` function, which is no longer used.

Available in Mac OS X v10.0 and later.

Declared in `LSInfo.h`.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

`kLSUnknownKindID`

A possible value of the `kindID` field of an item-information record, which is no longer used.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `LSInfo.h`.

`kLSMinCatInfoBitmap`

A minimal catalog information bitmap; no longer used.

Available in Mac OS X v10.1 and later.

Declared in `LSInfo.h`.

Result Codes

The table below lists the most common result codes returned by Launch Services functions.

Result Code	Value	Description
`kLSAppInTrashErr`	-10660	The application cannot be run because it is inside a Trash folder. Available in Mac OS X v10.3 and later.
`kLSUnknownErr`	-10810	An unknown error has occurred. Available in Mac OS X v10.0 and later.
`kLSNotAnApplicationErr`	-10811	The item to be registered is not an application. Available in Mac OS X v10.0 and later.
`kLSNotInitializedErr`	-10812	Formerly returned by `LSInit` on initialization failure; no longer used. Available in Mac OS X v10.0 and later.
`kLSDataUnavailableErr`	-10813	Data of the desired type is not available (for example, there is no kind string). Available in Mac OS X v10.0 and later.
`kLSApplicationNotFoundErr`	-10814	No application in the Launch Services database matches the input criteria. Available in Mac OS X v10.0 and later.
`kLSUnknownTypeErr`	-10815	Not currently used. Available in Mac OS X v10.0 and later.
`kLSDataTooOldErr`	-10816	Not currently used. Available in Mac OS X v10.0 and later.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Result Code	Value	Description
kLSDataErr	-10817	Data is structured improperly (for example, an item's information property list is malformed). Not used in Mac OS X v10.4. Available in Mac OS X v10.0 and later.
kLSLaunchInProgressErr	-10818	A launch of the application is already in progress. Available in Mac OS X v10.0 and later.
kLSNotRegisteredErr	-10819	Not currently used. Available in Mac OS X v10.0 and later.
kLSAppDoesNotClaimTypeErr	-10820	Not currently used. Available in Mac OS X v10.0 and later.
kLSAppDoesNotSupportSchemeWarning	-10821	Not currently used. Available in Mac OS X v10.0 and later.
kLSServerCommunicationErr	-10822	There is a problem communicating with the server process that maintains the Launch Services database. Available in Mac OS X v10.0 and later.
kLSCannotSetInfoErr	-10823	The filename extension to be hidden cannot be hidden. Available in Mac OS X v10.1 and later.
kLSNoRegistrationInfoErr	-10824	Not currently used. Available in Mac OS X v10.3 and later.
kLSIncompatibleSystemVersionErr	-10825	The application to be launched cannot run on the current Mac OS version. Available in Mac OS X v10.3 and later.
kLSNoLaunchPermissionErr	-10826	The user does not have permission to launch the application (on a managed network). Available in Mac OS X v10.3 and later.
kLSNoExecutableErr	-10827	The executable file is missing or has an unusable format. Available in Mac OS X v10.3 and later.
kLSNoClassicEnvironmentErr	-10828	The Classic emulation environment was required but is not available. Available in Mac OS X v10.3 and later.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Result Code	Value	Description
`kLSMultipleSessionsNotSupportedErr`	-10829	The application to be launched cannot run simultaneously in two different user sessions. Available in Mac OS X v10.3 and later.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Deprecated Launch Services Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.3

LSInit

(Deprecated in Mac OS X v10.3. Formerly used to initialize Launch Services; now does nothing.)

Not recommended.

```
OSStatus LSInit (
   LSInitializeFlags inFlags
);
```

Discussion
Calling this function was formerly required in order to initialize Launch Services; it is no longer needed, however, because Launch Services is now initialized automatically the first time one of its functions is called. `LSInit` now does nothing at all.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`LSInfo.h`

LSTerm

(Deprecated in Mac OS X v10.3. Formerly used to terminate Launch Services; now does nothing.)

Not recommended.

```
OSStatus LSTerm (
   void
);
```

Discussion
Calling this function was formerly required in order to terminate Launch Services; however, it is no longer needed and so should not be called. It now does nothing at all.

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Version Notes
Thread-safe since Mac OS version 10.2.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.

Declared In
`LSInfo.h`

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Document Revision History

This table describes the changes to *Launch Services Reference*.

Date	Notes
2006-07-13	Made minor formatting changes.
2006-01-10	Updated for Mac OS X v10.4.
	Added the following functions: `LSOpenApplication`,`LSOpenURLsWithRole`, `LSOpenItemsWithRole`, `LSCopyItemAttribute`, `LSCopyItemAttributes`, `LSCopyDefaultRoleHandlerForContentType`, `LSCopyAllRoleHandlersForContentType`, `LSSetDefaultRoleHandlerForContentType`, `LSGetHandlerOptionsForContentType`, `LSSetHandlerOptionsForContentType`, `LSCopyDefaultHandlerForURLScheme`, `LSCopyAllHandlersForURLScheme`, `LSSetDefaultHandlerForURLScheme`.
	Also documented the `LSApplicationParameters` structure, and added these constants: `kLSHandlerOptionsDefault`, `kLSHandlerOptionsIgnoreCreator`,`kLSItemContentType`, `kLSItemFileType`, `kLSItemFileCreator`, `kLSItemExtension`, `kLSItemDisplayName`, `kLSItemDisplayKind`, `kLSItemRoleHandlerDisplayName`, `kLSItemIsInvisible`, `kLSItemExtensionIsHidden` and `kLSLaunchHasUntrustedContents`.
2003-12-01	First version of *Launch Services Reference*

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Document Revision History

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

Index

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.

L

R

U

2006-07-13 | © 2003, 2006 Apple Computer, Inc. All Rights Reserved.