Launch Services Keys

Published: 2018-06-04 · Archived: 2026-04-05 21:57:24 UTC

Launch Services (part of the Core Services framework in macOS) provides support for launching apps and matching document types to apps. As a result, the keys recognized by Launch Services allow you to specify the desired execution environment for your bundled code. (In iOS, Launch Services is a private API but is still used internally to coordinate the execution environment of iOS apps.)

Launch Services keys use the prefix `LS` to distinguish them from other keys. For more information about Launch Services in macOS, see *Launch Services Programming Guide* and *Launch Services Reference*.

Key Summary

Table 1 contains an alphabetical listing of Launch Services keys, the corresponding name for that key in the Xcode property list editor, a high-level description of each key, and the platforms on which you use it. Detailed information about each key is available in later sections.

Table 1 Summary of Launch Services keys

Key	Xcode name	Summary	Platforms
LSApplicationCategoryType	“Application Category”	Contains a string with the UTI that categorizes the app for the App Store. See LSApplicationCategoryType for details.	macOS
LSApplicationQueriesSchemes	None	Specifies the URL schemes the app is able to test using the `canOpenURL:` method. See LSApplicationQueriesSchemes for details.	iOS 9.0 and later
LSArchitecturePriority	“Architecture priority”	Contains an array of strings identifying the supported code architectures and their preferred execution priority. See LSArchitecturePriority for details.	macOS
LSBackgroundOnly	“Application is background only”	Specifies whether the app runs only in the background. (Mach-O apps only). See LSBackgroundOnly for details.	macOS

LSEnvironment	“Environment variables”	Contains a list of key/value pairs, representing environment variables and their values. See LSEnvironment for details.	macOS
LSFileQuarantineEnabled	“File quarantine enabled”	Specifies whether the files this app creates are quarantined by default. See LSFileQuarantineEnabled.	macOS
LSFileQuarantineExcludedPathPatterns	None	Specifies directories for which files should not be automatically quarantined. See LSFileQuarantineExcludedPathPatterns.	macOS
LSGetAppDiedEvents	“Application should get App Died events”	Specifies whether the app is notified when a child process dies. See LSGetAppDiedEvents for details.	macOS
LSMinimumSystemVersion	“Minimum system version”	Specifies the minimum version of macOS required for the app to run. See LSMinimumSystemVersion for details.	macOS
LSMinimumSystemVersionByArchitecture	“Minimum system versions, per-architecture”	Specifies the minimum version of macOS required to run a given platform architecture. See LSMinimumSystemVersionByArchitecture for details.	macOS
LSMultipleInstancesProhibited	“Application prohibits multiple instances”	Specifies whether one user or multiple users can launch an app simultaneously. See LSMultipleInstancesProhibited for details.	macOS
LSRequiresIPhoneOS	“Application requires iPhone environment”	Specifies whether the app is an iOS app. See LSRequiresIPhoneOS for details.	iOS

LSRequiresNativeExecution	“Application requires native environment”	Specifies whether the app must run natively on an Intel-based Mac, as opposed to under Rosetta emulation. See LSRequiresNativeExecution for details.	macOS
LSSupportsOpeningDocumentsInPlace	None	Enables your app to directly open original documents, not copies, from other apps. See LSSupportsOpeningDocumentsInPlace for details.	iOS
LSUIElement	“Application is agent (UIElement)”	Specifies whether the app is an agent app, that is, an app that should not appear in the Dock or Force Quit window. See LSUIElement for details.	macOS
LSUIPresentationMode	“Application UI Presentation Mode”	Sets the visibility of system UI elements when the app launches. See LSUIPresentationMode for details.	macOS
LSVisibleInClassic	“Application is visible in Classic”	Specifies whether an agent app or background-only app is visible to other apps in the Classic environment. See LSVisibleInClassic for details.	macOS
MinimumOSVersion	“Minimum system version”	Do not use. Use LSMinimumSystemVersion instead.	iOS

LSApplicationCategoryType

`LSApplicationCategoryType` (`String` - macOS) is a string that contains the UTI corresponding to the app’s type. The App Store uses this string to determine the appropriate categorization for the app. Table 2 lists the supported UTIs for apps.

Table 2 UTIs for app categories

Category	**UTI**

Business	`public.app-category.business`
Developer Tools	`public.app-category.developer-tools`
Education	`public.app-category.education`
Entertainment	`public.app-category.entertainment`
Finance	`public.app-category.finance`
Games	`public.app-category.games`
Graphics & Design	`public.app-category.graphics-design`
Healthcare & Fitness	`public.app-category.healthcare-fitness`
Lifestyle	`public.app-category.lifestyle`
Medical	`public.app-category.medical`
Music	`public.app-category.music`
News	`public.app-category.news`
Photography	`public.app-category.photography`
Productivity	`public.app-category.productivity`
Reference	`public.app-category.reference`
Social Networking	`public.app-category.social-networking`

Sports	`public.app-category.sports`
Travel	`public.app-category.travel`
Utilities	`public.app-category.utilities`
Video	`public.app-category.video`
Weather	`public.app-category.weather`

Table 3 lists the UTIs that are specific to games.

Table 3 UTIs for game-specific categories

Category	**UTI**
Action Games	`public.app-category.action-games`
Adventure Games	`public.app-category.adventure-games`
Arcade Games	`public.app-category.arcade-games`
Board Games	`public.app-category.board-games`
Card Games	`public.app-category.card-games`
Casino Games	`public.app-category.casino-games`
Dice Games	`public.app-category.dice-games`
Educational Games	`public.app-category.educational-games`

Family Games	`public.app-category.family-games`
Kids Games	`public.app-category.kids-games`
Music Games	`public.app-category.music-games`
Puzzle Games	`public.app-category.puzzle-games`
Racing Games	`public.app-category.racing-games`
Role Playing Games	`public.app-category.role-playing-games`
Simulation Games	`public.app-category.simulation-games`
Sports Games	`public.app-category.sports-games`
Strategy Games	`public.app-category.strategy-games`
Trivia Games	`public.app-category.trivia-games`
Word Games	`public.app-category.word-games`

LSApplicationQueriesSchemes

`LSApplicationQueriesSchemes` (`Array` - iOS) Specifies the URL schemes you want the app to be able to use with the `canOpenURL:` method of the `UIApplication` class. For each URL scheme you want your app to use with the `canOpenURL:` method, add it as a string in this array. Read the `canOpenURL:` method description for important information about declaring supported schemes and using that method.

To learn about the converse operation of registering the URL schemes an app can handle, read the description of the CFBundleURLTypes key.

This key is supported in iOS 9.0 and later.

LSArchitecturePriority

`LSArchitecturePriority` (`Array` - macOS) is an array of strings that identifies the architectures this app supports. The order of the strings in this array dictates the preferred execution priority for the architectures. The possible strings for this array are listed in Table 4.

Table 4 Execution architecture identifiers

String	Description
`i386`	The 32-bit Intel architecture.
`ppc`	The 32-bit PowerPC architecture.
`x86_64`	The 64-bit Intel architecture.
`ppc64`	The 64-bit PowerPC architecture.

If a PowerPC architecture appears before either of the Intel architectures, macOS runs the executable under Rosetta emulation on an Intel-based Mac. To force macOS to use the current platform's native architecture, include the LSRequiresNativeExecution key in your information property list.

LSBackgroundOnly

`LSBackgroundOnly` (`Boolean` - macOS) specifies whether this app runs only in the background. If this key exists and is set to `YES` , Launch Services runs the app in the background only. You can use this key to create faceless background apps. You should also use this key if your app uses higher-level frameworks that connect to the window server, but are not intended to be visible to users. Background apps must be compiled as Mach-O executables. This option is not available for CFM apps.

LSEnvironment

`LSEnvironment` (`Dictionary` - macOS) defines environment variables to be set before launching this app. The names of the environment variables are the keys of the dictionary, with the values being the corresponding environment variable value. Both keys and values must be strings.

These environment variables are set only for apps launched through Launch Services. If you run your executable directly from the command line, these environment variables are not set.

LSFileQuarantineEnabled

`LSFileQuarantineEnabled` (`Boolean` - macOS) specifies whether files this app creates are quarantined by default.

Value	Description

`true`	Files created by this app are quarantined by default. When quarantining files, the system automatically associates the timestamp, app name, and the bundle identifier with the quarantined file whenever possible. Your app can also get or set quarantine attributes as needed using Launch Services.
`false`	(Default) Files created by this app are not quarantined by default.

This key is available in macOS 10.5 and later.

LSFileQuarantineExcludedPathPatterns

`LSFileQuarantineExcludedPathPatterns` (`Array` - macOS) contains an array of strings indicating the paths for which you want to disable file quarantining. You can use this key to prevent file quarantines from affecting the performance of your app. Each string in the array is a shell-style path pattern, which means that special characters such as `~` , `*` , and `?` are automatically expanded according to the standard command-line rules. For example, a string of the form `~/Library/Caches/*` would allow you to disable the quarantine for files created by your app in the user's cache directory.

LSGetAppDiedEvents

`LSGetAppDiedEvents` (`Boolean` - macOS) indicates whether the operation system notifies this app when when one of its child process terminates. If you set the value of this key to `YES` , the system sends your app an `kAEApplicationDied` Apple event for each child process as it terminates.

LSMinimumSystemVersion

`LSMinimumSystemVersion` (`String` - macOS) indicates the minimum version of macOS required for this app to run. This string must be of the form *n.n.n* where *n* is a number. The first number is the major version number of the system. The second and third numbers are minor revision numbers. For example, to support macOS v10.4 and later, you would set the value of this key to "10.4.0".

If the minimum system version is not available, macOS tries to display an alert panel notifying the user of that fact.

LSMinimumSystemVersionByArchitecture

`LSMinimumSystemVersionByArchitecture` (`Dictionary` - macOS) specifies the earliest macOS version for a set of architectures. This key contains a dictionary of key-value pairs. Each key corresponds to one of the architectures associated with the LSArchitecturePriority key. The value for each key is the minimum version of macOS required for the app to run under that architecture. This string must be of the form *n.n.n* where *n* is a number. The first number is the major version number of the system. The second and third numbers are minor revision numbers. For example, to support macOS 10.4.9 and later, you would set the value of this key to "10.4.9".

If the current system version is less than the required minimum version, Launch Services does not attempt to use the corresponding architecture. This key applies only to the selection of an execution architecture and can be used in conjunction with the LSMinimumSystemVersion key, which specifies the overall minimum system version requirement for the app.

LSMultipleInstancesProhibited

`LSMultipleInstancesProhibited` (`Boolean` - macOS) indicates whether an app is prohibited from running simultaneously in multiple user sessions. If true, the app runs in only one user session at a time. You can use this key to prevent resource conflicts that might arise by sharing an app across multiple user sessions. For example, you might want to prevent users from accessing a custom USB device when it is already in use by a different user.

Launch Services returns an appropriate error code if the target app cannot be launched. If a user in another session is running the app, Launch Services returns a `kLSMultipleSessionsNotSupportedErr` error. If you attempt to launch a separate instance of an app in the current session, it returns `kLSMultipleInstancesProhibitedErr`.

LSRequiresIPhoneOS

`LSRequiresIPhoneOS` (`Boolean` - iOS) specifies whether the app can run only on iOS. If this key is set to `YES`, Launch Services allows the app to launch only when the host platform is iOS.

LSRequiresNativeExecution

`LSRequiresNativeExecution` (`Boolean` - macOS) specifies whether to launch the app using the subbinary for the current architecture. If this key is set to `YES`, Launch Services always runs the app using the binary compiled for the current architecture. You can use this key to prevent a universal binary from being run under Rosetta emulation on an Intel-based Mac. For more information about configuring the execution architectures, see LSArchitecturePriority.

LSSupportsOpeningDocumentsInPlace

`LSSupportsOpeningDocumentsInPlace` (`Boolean` - iOS) When set to a value of `YES`, enables your app to open the *original* document from a file provider, rather than a *copy* of the document. The app can access documents from the system's local file provider, the iCloud file provider, and any third-party File Provider extensions that support opening documents in place.

The URL for a document opened in place is security-scoped. For information about working with security-scoped URLs and bookmarks, read the overview in *NSURL Class Reference* and read Document Provider in *App Extension Programming Guide*.

In iOS 11 and later, if both this key and the UIFileSharingEnabled key are `YES`, the local file provider grants access to all the documents in the app's `Documents` directory. These documents appear in the Files app, and in a document browser. Users can open and edit these document in place.

LSUIElement

`LSUIElement` (`Boolean` - macOS) specifies whether the app runs as an agent app. If this key is set to `YES`, Launch Services runs the app as an agent app. Agent apps do not appear in the Dock or in the Force Quit window. Although they typically run as background apps, they can come to the foreground to present a user interface if desired. A click on a window belonging to an agent app brings that app forward to handle events.

The Dock and loginwindow are two apps that run as agent apps.

LSUIPresentationMode

`LSUIPresentationMode` (`Number` - macOS) identifies the initial user-interface mode for the app. You would use this in apps that may need to take over portions of the screen that contain UI elements such as the Dock and menu bar. Most modes affect only UI elements that appear in the content area of the screen, that is, the area of the screen that does not include the menu bar. However, you can request that all UI elements be hidden as well.

This key is applicable to both Carbon and Cocoa apps and can be one of the following values:

Value	Description
0	Normal mode. In this mode, all standard system UI elements are visible. This is the default value.
1	Content suppressed mode. In this mode, system UI elements in the content area of the screen are hidden. UI elements may show themselves automatically in response to mouse movements or other user activity. For example, the Dock may show itself when the mouse moves into the Dock's auto-show region.
2	Content hidden mode. In this mode, system UI elements in the content area of the screen are hidden and do not automatically show themselves in response to mouse movements or user activity.
3	All hidden mode. In this mode, all UI elements are hidden, including the menu bar. Elements do not automatically show themselves in response to mouse movements or user activity.
4	All suppressed mode. In this mode, all UI elements are hidden, including the menu bar. UI elements may show themselves automatically in response to mouse movements or other user activity. This option is available only in macOS v10.3 and later.

LSVisibleInClassic

`LSVisibleInClassic` (`String` - macOS). If this key is set to "1", any agent apps or background-only apps with this key appears as background-only processes to the Classic environment. Agent apps and background-only apps without this key do not appear as running processes to Classic at all. Unless your process needs to communicate explicitly with a Classic app, you do not need to include this key.

MinimumOSVersion

`MinimumOSVersion` (`String` - iOS). When you build an iOS app, Xcode uses the iOS Deployment Target setting of the project to set the value for the `MinimumOSVersion` key. Do *not* specify this key yourself in the `Info.plist` file; it is a system-written key. When you publish your app to the App Store, the store indicates the iOS releases on which your app can run based on this key. It is equivalent to the `LSMinimumSystemVersion` key in macOS.
