Entitlement Key Reference

Developer

Contents

Tables

About Entitlements

Entitlements confer specific capabilities or security permissions to your iOS or OS X app.

At a Glance

Set entitlement values in order to enable iCloud, push notifications, and App Sandbox. Each entitlement has a default value, which in most cases disables the capability associated with the entitlement. When you set an entitlement, you are overriding the default by providing an appropriate key-value pair.

- iCloud entitlements let you enable the use of iCloud data storage for your iOS or OS X app.

  You set iCloud entitlement values on a target-by-target basis in your Xcode project.

- Push notifications let your app alert the user even when your iOS or OS X app is not executing.

  You set push notification entitlement values as part of configuring your development and distribution provisioning profiles.

- App Sandbox entitlements let you enable the security feature called *sandboxing* for your OS X app. (In iOS, all apps are sandboxed automatically, so these sandboxing entitlements do not apply.)

  By carefully enabling only the resource access that you need, you minimize the potential for damage if malicious code successfully exploits your app. You set App Sandbox entitlement values on a target-by-target basis in your Xcode project.

You can set many entitlements using the Summary tab of the Xcode target editor. Other entitlements require editing a target's entitlements property list file. Finally, a few entitlements are inherited from the iOS provisioning profile used to run the app.

The sort of value to associate with an entitlement key depends on the key. Many entitlement keys take Boolean values. For entitlements defined in a property list in an Xcode project, a Boolean entitlement value is either `<true/>` or `<false/>`. Some entitlement keys take a string or an array of strings as a value. Refer to the chapters in this document for specifics on the values to apply to the various entitlement keys.

To use any entitlement keys, you must code sign your app because an app's entitlements are built in to its code signature.

Enable iCloud for Sharing Data Among Devices

Xcode's target editor contains two fields that let you enable iCloud document and key-value storage for your app.

Relevant chapter: Enabling iCloud Storage (page 8)

Enable Push Notifications for Alerting the User

You can send push notifications to users, by way of the Apple Push Notification service (APNs), to let users know your app has information for them. To enable the receiving of such notifications in your app, request the appropriate entitlement within your development and distribution provisioning profiles.

Relevant chapter: Enabling Push Notifications (page 10)

Enable App Sandbox to Minimize Damage from Malicious Code

Employ Xcode's target editor to turn on and configure App Sandbox for targets in an OS X project.

Relevant chapters: Enabling App Sandbox (page 11)

Use App Sandbox Temporary Exceptions, If Needed

If you are unable to transition your entire app to App Sandbox in a single release, you can employ special temporary-exception entitlements.

Relevant chapter: App Sandbox Temporary Exception Entitlements (page 26)

Prerequisites

Understand where entitlements fit into the development process by reading *App Programming Guide for iOS* or *Mac App Programming Guide*.

You set iCloud and App Sandbox entitlement values using Xcode. For an introduction to Xcode, read *Xcode Overview*.

See Also

When adding iCloud features to your app, be sure to read *iCloud Design Guide*.

When adding push notification capability to your app, refer to *Local and Remote Notification Programming Guide*.

When configuring your sandbox, use this document in concert with *App Sandbox Design Guide*.

Enabling iCloud Storage

To enable your app to use iCloud document or key-value storage, set values for the appropriate entitlements using Xcode.

iCloud Entitlement Keys

The following table describes the entitlement keys that you set values for to enable iCloud storage:

iCloud entitlement key	Capability
`com.apple.developer.ubiquity-container-identifiers`	iCloud document storage
`com.apple.developer.ubiquity-kvstore-identifier`	iCloud key-value storage

Enabling iCloud Document Storage

To support iCloud storage of user documents, use the iCloud Containers setting in the Xcode target editor.

As the value for this entitlement, provide an array of one or more strings. One of these strings must be the bundle identifier for your app, or for another app that you submit using the same team identifier. In the `.entitlements` file, this string has the following form:

```
$(TeamIdentifierPrefix)com.mycompany.myapplication
```

If you are using the graphical interface of the Xcode target editor to provide a value for the iCloud container entitlement, there is no need to include the `$(TeamIdentifierPrefix)` variable. Xcode supplies the prefix in the `.entitlements` file automatically.

If you want to confer access to documents created by other apps published by your team, use additional strings to specify the bundle identifiers of those apps.

You must not use a wildcard (“*”) character in the string for an iCloud container entitlement value.

Table 1-1 Xcode setting for iCloud document storage

Setting	Entitlement key
iCloud Containers	`com.apple.developer.ubiquity-container-identifiers`

Enabling iCloud Key-Value Storage

To support iCloud storage of key-value information for your app, use the iCloud Key-Value Store setting in the Xcode target editor.

As the value for this entitlement key, provide the bundle identifier for your app, such as:

```
$(TeamIdentifierPrefix)com.mycompany.myapplication
```

If you are using the graphical interface of the target editor to provide a value for the iCloud key-value store entitlement, there is no need to include the `$(TeamIdentifierPrefix)` variable. Xcode supplies the prefix in the `.entitlements` file automatically.

Table 1-2 Xcode setting for iCloud key-value storage

Setting	Entitlement key
iCloud Key-Value Store	`com.apple.developer.ubiquity-kvstore-identifier`

Enabling Push Notifications

To enable your app to alert the user through push (also known as *remote*) notifications, set a value for the appropriate entitlement.

> **Note:** Push notifications have no functional connection to broadcast notifications or key-value observing notifications.

For a complete discussion about push notifications, see *Local and Remote Notification Programming Guide*.

Push Notification Entitlement Keys

You set a value for the push notification entitlement by way of your development and distribution provisioning profiles, as described in *Local and Remote Notification Programming Guide*.

The following table shows the push notification entitlement keys that apply to the iOS and OS X platforms:

Notification entitlement key	Capability
`aps-environment`	Receive push notifications in iOS
`com.apple.developer.aps-environment`	Receive push notifications in OS X

The entitlement key is different for iOS than it is for OS X. On either platform, however, the provisioning portal assigns a value of `development` or `production` to the key, depending only on which activity you are creating the provisioning profile for.

Enabling App Sandbox

Note: This chapter describes property list keys specific to the OS X implementation of App Sandbox. They are not available in iOS.

In your OS X Xcode project, configure fine-grained security permissions by enabling settings in the Summary tab of the target editor. These settings, in turn, add Boolean values to entitlement keys in the target's `.entitlements` property list file. The values are then incorporated into the target's code signature when you build the project.

You can think of using App Sandbox entitlements as a two-step process:

1. Sandbox a target, which removes most capabilities for interacting with the system
2. Restore capabilities to the sandboxed target, as needed, by configuring App Sandbox entitlements

At runtime, if a target requires a capability or a system resource for which the target isn't entitled, the sandbox daemon (`sandboxd`) logs a violation message to the console.

For more information about App Sandbox, read *App Sandbox Design Guide*.

App Sandbox Entitlement Keys

This section describes the keys you can use to confer capabilities to a sandboxed app in OS X. The first key enables App Sandbox; the others configure the sandbox. If App Sandbox is not enabled, the other keys in this section are meaningless.

The value to use for any of these keys is a Boolean `YES` or `NO`, with the default value in each case being `NO`. If you are editing the `.entitlements` file directly in a text editor, the corresponding Boolean values to use are `<true/>` and `<false/>`. The default value for each key is false, so you can (and generally should) leave out the entitlement entirely rather than specifying a false value.

In cases where there are read-only and read/write entitlement key pairs, use of either key in the pair is mutually exclusive with the other.

Add these keys by using the Summary tab of the Xcode target editor. You can also add them directly to a target's `.entitlements` file with the Xcode property list editor.

For information on additional entitlements for handling special circumstances, see App Sandbox Temporary Exception Entitlements (page 26).

For each key in this table, providing a Boolean value of YES enables the corresponding capability (unless otherwise noted).

Entitlement key	Capability
`com.apple.security.app-sandbox`	Enables App Sandbox for a target in an Xcode project
`com.apple.security.application-groups`	Allows access to group containers that are shared among multiple apps produced by a single development team, and allows certain additional interprocess communication between the apps Supported in OS X v10.7.5 and in v10.8.3 and later. The format for this attribute is described in Adding an App to an App Group (page 22).
`com.apple.security.assets.movies.read-only`	Read-only access to the user's Movies folder and iTunes movies For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.assets.movies.read-write`	Read/write access to the user's Movies folder and iTunes movies For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.assets.music.read-only`	Read-only access to the user's Music folder For details, see Enabling Access to Files in Standard Locations (page 18).

Entitlement key	Capability
`com.apple.security.assets.music.read-write`	Read/write access to the user's Music folder For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.assets.pictures.read-only`	Read-only access to the user's Pictures folder For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.assets.pictures.read-write`	Read/write access to the user's Pictures folder For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.device.audio-video-bridging`	Communication with AVB devices For details, see Enabling Hardware Access (page 20).
`com.apple.security.device.bluetooth`	Interaction with Bluetooth devices For details, see Enabling Hardware Access (page 20).
`com.apple.security.device.camera`	Capture of movies and still images using the built-in camera, if available For details, see Enabling Hardware Access (page 20).
`com.apple.security.device.firewire`	Interaction with FireWire devices (currently, does not support interaction with audio/video devices such as DV cameras) For details, see Enabling Hardware Access (page 20).

Entitlement key	Capability
`com.apple.security.device.microphone`	Recording of audio using the built-in microphone, if available, along with access to audio input using any Core Audio API that supports audio input For details, see Enabling Hardware Access (page 20).
`com.apple.security.device.serial`	Interaction with serial devices For details, see Enabling Hardware Access (page 20).
`com.apple.security.device.usb`	Interaction with USB devices, including HID devices such as joysticks For details, see Enabling Hardware Access (page 20).
`com.apple.security.files.downloads.read-write`	Read/write access to the user's Downloads folder For details, see Enabling Access to Files in Standard Locations (page 18).
`com.apple.security.files.bookmarks.app-scope`	Use of app-scoped bookmarks and URLs For details, see Enabling Security-Scoped Bookmark and URL Access (page 19).
`com.apple.security.files.bookmarks.document-scope`	Use of document-scoped bookmarks and URLs For details, see Enabling Security-Scoped Bookmark and URL Access (page 19).

Entitlement key	Capability
`com.apple.security.files.user-selected.read-only`	Read-only access to files the user has selected using an Open or Save dialog For details, see Enabling User-Selected File Access (page 17).
`com.apple.security.files.user-selected.read-write`	Read/write access to files the user has selected using an Open or Save dialog For details, see Enabling User-Selected File Access (page 17).
`com.apple.security.files.user-selected.executable`	Allows apps to write executable files. For details, see Enabling User-Selected File Access (page 17).
`com.apple.security.inherit`	Child process inheritance of the parent's sandbox For details, see Enabling App Sandbox Inheritance (page 23).
`com.apple.security.network.client`	Network socket for connecting to other machines For details, see Enabling Network Access (page 20).
`com.apple.security.network.server`	Network socket for listening for incoming connections initiated by other machines For details, see Enabling Network Access (page 20).

Entitlement key	Capability
`com.apple.security.personal-information.addressbook`	Read/write access to contacts in the user's address book; allows apps to infer the default address book if more than one is present on a system For details, see Enabling Personal Information Access (page 22).
`com.apple.security.personal-information.calendars`	Read/write access to the user's calendars For details, see Enabling Personal Information Access (page 22).
`com.apple.security.personal-information.location`	Use of the Core Location framework for determining the computer's geographical location For details, see Enabling Personal Information Access (page 22).
`com.apple.security.print`	Printing For details, see Enabling Hardware Access (page 20).
`com.apple.security.scripting-targets`	Ability to use specific AppleScript scripting access groups within a specific scriptable app For details, see Enabling Scripting of Other Apps (page 24).

Enabling App Sandbox

You enable App Sandbox individually for each target in an OS X Xcode project. For example, you may design a project as a main app, and some helpers in the form of XPC services. You then enable and configure the sandbox for each target individually.

To learn how to enable App Sandbox for your OS X app, which includes performing code signing, see App Sandbox Quick Start in *App Sandbox Design Guide* . The essential step is to ensure that the target editor checkbox named in Table 3-1 is selected.

Table 3-1 Xcode setting for enabling App Sandbox

Setting	Entitlement key
Enable App Sandboxing	`com.apple.security.app-sandbox`

Enabling User-Selected File Access

Xcode provides a pop-up menu, in the Summary tab of the target editor, with choices to enable read-only or read/write access to files and folders that the user explicitly selects. When you enable user-selected file access, you gain programmatic access to files and folders that the user opens using an `NSOpenPanel` object, and files the user saves using an `NSSavePanel` object.

Certain other user interactions, such as dragging items to your app or choosing items from the Open Recent menu, automatically expand your sandbox to include those items. Similarly, when OS X resumes an app after a reboot, the sandbox is automatically expanded to include any items that are automatically opened.

To enable user-selected file access in your app, use the Xcode target editor setting shown in Table 3-2.

Note: If your app needs to create executable files that are typically executed in some way other than through Launch Services (shell scripts, for example), you should also specify the `com.apple.security.files.user-selected.executable` entitlement.

By default, when writing executable files in sandboxed apps, the files are quarantined. Gatekeeper prevents quarantined executable files and other similar files (shell scripts, web archives, and so on) from opening or executing unless the user explicitly launches them from Finder.

If those executables are tools that are intended to run from the command line, such as shell scripts, this presents a problem. With this flag, the file quarantine system allows the app to write non-quarantined executables so that Gatekeeper does not prevent them from executing.

This entitlement does not have an Xcode checkbox, and thus must be added to your app's entitlement property list manually. For details, see App Sandbox Entitlement Keys (page 11).

Table 3-2 Xcode setting for user-selected file and folder access

Setting	Entitlement keys
User Selected File	`com.apple.security.files.user-selected.read-only` `com.apple.security.files.user-selected.read-write`

Enabling Access to Files in Standard Locations

In addition to granting user-selected file access, you can employ entitlements to grant programmatic file access to the following user folders:

- Downloads
- Music
- Movies
- Pictures

The Xcode control for enabling Downloads folder access is a checkbox; the controls for enabling access to these other folders are pop-up menus.

When you enable programmatic access to the user's Movies folder, you also gain access to their iTunes movies.

Reopening of files by OS X using Resume does not require the presence of any entitlement key.

To enable programmatic access to specific folders, use the Xcode target editor settings shown in Table 3-3.

Table 3-3 Xcode settings for programmatic file and folder access

Setting	Entitlement keys
Downloads Folder	`com.apple.security.files.downloads.read-write`
Music Folder	`com.apple.security.assets.music.read-only` `com.apple.security.assets.music.read-write`
Movies Folder	`com.apple.security.assets.movies.read-only` `com.apple.security.assets.movies.read-write`
Pictures Folder	`com.apple.security.assets.pictures.read-only` `com.apple.security.assets.pictures.read-write`

Enabling Security-Scoped Bookmark and URL Access

If you want to provide your sandboxed app with persistent access to file system resources, you must enable security-scoped bookmark and URL access. Security-scoped bookmarks are available starting in OS X v10.7.3.

To add the `bookmarks.app-scope` or `bookmarks.document-scope` entitlement, edit the target's `.entitlements` property list file using the Xcode property list editor. Use the entitlement keys shown in Table 3-4, depending on which type of access you want. Use a value of `<true/>` for each entitlement you want to enable. You can enable either or both entitlements.

For more information on security-scoped bookmarks, read Security-Scoped Bookmarks and Persistent Resource Access in *App Sandbox Design Guide*.

Table 3-4 Entitlement keys for enabling security-scoped bookmark and URL access

Entitlement key	Description
`com.apple.security.files.bookmarks.app-scope`	Enables use of app-scoped bookmarks and URLs
`com.apple.security.files.bookmarks.document-scope`	Enables use of document-scoped bookmarks and Version note: in OS X v10.7.3, this entitlement key v `com.apple.security.files.bookmarks.col`

Enabling Network Access

Xcode's Network checkboxes in the Summary tab of the target editor let you enable network access for your app.

To enable your app to connect to a server process running on another machine (or on the same machine), enable outgoing network connections.

To enable opening a network listening socket so that other computers can connect to your app, allow incoming network connections.

> **Note:** Both outgoing and incoming connections can send and receive data. The sole difference is in whether your app is initiating the connection or is receiving connections initiated by other apps or other hosts.

To enable network access, use the Xcode target editor settings shown in Table 3-5.

Table 3-5 Xcode settings for network access

Setting	Entitlement key
Allow Incoming Connections	`com.apple.security.network.server`
Allow Outgoing Connections	`com.apple.security.network.client`

Enabling Hardware Access

To allow a sandboxed target to access hardware services on a system—USB, printing, or the built-in camera and microphone—enable the corresponding setting in the Summary tab of the Xcode target editor.

- *Camera access* enables access to video and still image capture using the built-in camera, if available.
- *Microphone access* enables access to audio recording using the built-in microphone, if available.
- *USB access* enables the ability to interact with USB devices using USB device access APIs. On violation, `sandboxd` names the I/O Kit class your code tried to access.
- *Printing access* is required if you want to provide a target with the ability to print.

To enable access to hardware, use the Xcode target editor settings shown in Table 3-6.

Table 3-6 Xcode settings for hardware access

Setting	Entitlement key
Allow Camera Access	`com.apple.security.device.camera`
Allow Microphone Access	`com.apple.security.device.microphone`
Allow USB Access	`com.apple.security.device.usb`
Allow Printing	`com.apple.security.print`

To allow access to hardware devices for which no checkbox exists in Xcode's user interface, you must manually add the appropriate entitlement to your app's entitlements property list. These additional entitlements are listed in Table 3-7 (page 21). All of these keys take a Boolean value.

Table 3-7 Other entitlement keys for accessing hardware

Entitlement key	Description
`com.apple.security.device.audio-video-bridging`	Interaction with AVB devices by using the Audio Video Bridging framework
`com.apple.security.device.bluetooth`	Interaction with Bluetooth devices
`com.apple.security.device.firewire`	Interaction with FireWire devices (currently, does not support interaction with audio/video devices such as DV cameras)
`com.apple.security.device.serial`	Interaction with serial devices

Enabling Personal Information Access

A user's personal information is inaccessible to your sandboxed app unless you grant access using the appropriate settings.

- *Address Book* access enables read/write access to contacts in the user's address book.
- *Location Services* access enables use of the Core Location framework to determine the computer's geographic position.
- *Calendar access* enables read/write access to the user's calendars.

To enable access to personal information, use the Xcode target editor settings shown in Table 3-8.

Table 3-8 Xcode settings for access to a user's personal information

Setting	Entitlement key
Allow Address Book Data Access	`com.apple.security.personal-information.addressbook`
Allow Location Services Access	`com.apple.security.personal-information.location`
Allow Calendar Data Access	`com.apple.security.personal-information.calendars`

Adding an App to an App Group

The `com.apple.security.application-groups` (available in OS X v10.7.5 and v10.8.3 and later) allows multiple apps produced by a single development team to share access to a special group container. This container is intended for content that is not user-facing, such as shared caches or databases.

In addition, this attribute allows the apps within the group to share Mach and POSIX semaphores and to use certain other IPC mechanisms among the group's members. For additional details and naming conventions, read "Mach IPC and POSIX Semaphores and Shared Memory" in *App Sandbox Design Guide*.

The value for this key must be of type `array`, and must contain one or more `string` values, each of which must consist of your development team ID, followed by a period, followed by an arbitrary name chosen by your development team. For example:

```
<key>com.apple.security.application-groups</key>
<array>
    <string>DG29478A379Q6483R9214.HolstFirstAppSuite</string>
    <string>DG29478A379Q6483R9214.HolstSecondAppSuite</string>
```

```
</array>
```

The group containers are automatically created or added into each app's sandbox container as determined by the existence of these keys. The group containers are stored in `~/Library/Group Containers/<application-group-id>`, where `<application-group-id>` is one of the strings from the array. Your app can obtain the path to the group containers by calling the `containerURLForSecurityApplicationGroupIdentifier:` method of `NSFileManager`.

Enabling App Sandbox Inheritance

If your app employs a child process created with either the `posix_spawn` function or the `NSTask` class, you can configure the child process to inherit the sandbox of its parent. However, using a child process does not provide the security afforded by using an XPC service.

> **Important:** XPC (as described in External Tools, XPC Services, and Privilege Separation) complements App Sandbox and is the preferred technology for implementing privilege separation in an OS X app. Before using a child process, consider using an XPC service instead.

To enable sandbox inheritance, a child target must use exactly two App Sandbox entitlement keys: `com.apple.security.app-sandbox` and `com.apple.security.inherit`. If you specify any other App Sandbox entitlement, the system aborts the child process. You can, however, confer other capabilities to a child process by way of iCloud and notification entitlements.

The main app in an Xcode project must never have a `YES` value for the `inherit` entitlement.

To add the `inherit` entitlement, edit the target's `.entitlements` property list file using the Xcode property list editor. Use the entitlement key shown in Table 3-9 with a value of `<true/>`.

Note: This property causes the child process to inherit *only* the static rights defined in the main app's entitlements file, *not* any rights added to your sandbox after launch (such as PowerBox access to files).

If you need to provide access to files opened after launch, you must either pass the data to the helper or pass a bookmark to the child process. The bookmark need not be a security-scoped bookmark, but it can be, if desired.

If you are using other APIs to create child processes (such as `NSWorkspace`) and wish to have a shared container directory, you must use an app group.

Table 3-9 Entitlement key for inheriting the parent process's App Sandbox

Entitlement key	Description
`com.apple.security.inherit`	Enables App Sandbox inheritance

Enabling Scripting of Other Apps

If your app needs to control another scriptable app, your app can use the scripting targets entitlement to request access to one or more of the scriptable app's scripting access groups.

Note: Before you can use this entitlement, the scriptable app must provide scripting access groups. If it does not, you can still control the app, but you must use a temporary exception, as described in Apple Event Temporary Exception (page 26).

Table 3-10 Entitlement key for accessing scripting targets

Entitlement key	Description
`com.apple.security.scripting-targets`	Ability to use specific AppleScript scripting access groups within a specific scriptable app

The scripting target entitlement contains a dictionary where each entry has the target app's code signing identifier as the key, and an array of scripting access groups as the value. Scripting access groups are identified by strings and are specific to an app. For example, the following entry would grant access to composing mail messages with Apple's Mail app:

```
<key>com.apple.security.scripting-targets</key>
    <dict>
        <key>com.apple.mail</key>
        <array>
            <string>com.apple.mail.compose</string>
        </array>
    </dict>
```

For more information about how to add scripting access groups to an app, watch *WWDC 2012: Secure Automation Techniques in OS X* and read the manual page for `sdef`.

App Sandbox Temporary Exception Entitlements

> **Note:** This chapter describes property list keys specific to the OS X implementation of App Sandbox. They are not available in iOS.

A temporary exception entitlement permits your OS X app to perform certain operations otherwise disallowed by App Sandbox.

If you need to request a temporary exception entitlement, use Apple's bug reporting system to let Apple know what's not working for you. Apple considers feature requests as it develops the OS X platform.

> **Note:** If you request a temporary-exception entitlement, be sure to follow the guidance regarding entitlements provided on the iTunes Connect website. In particular, identify the entitlement and corresponding issue number in the App Sandbox Entitlement Usage Information section in iTunes Connect and explain why your app needs the exception.

To request a temporary exception entitlement for a target in an OS X Xcode project, add it to the target's `.entitlements` property list file using the Xcode property list editor.

The value to provide for any temporary exception entitlement is a string or an array of one or more strings. For more information on using temporary exceptions in OS X, refer to Designing for App Sandbox in *App Sandbox Design Guide*.

Apple Event Temporary Exception

When you adopt App Sandbox, your app retains the ability to:

- Receive Apple events
- Send Apple events to itself
- Respond to Apple events it receives

However, with App Sandbox you cannot send Apple events to other apps unless you configure a `scripting-targets` entitlement or an `apple-events` temporary exception entitlement.

The `scripting-targets` entitlement is the preferred way to request the ability to send Apple events to apps that provide scripting access groups. This entitlement is described in App Sandbox Entitlement Keys (page 11).

This entitlement contains an array of strings, each of which should contain the bundle identifier of an app you want to send Apple events to, with all characters in the bundle identifier converted to lowercase. For example, to enable sending Apple events to iPhoto from your app, you would pass an array containing a single string whose value is `com.apple.iphoto`.

Entitlement key	Capability
`com.apple.security.temporary-exception.apple-events`	Enables sending of Apple events to one or more destination apps.

Audio Unit Hosting Temporary Exception

By default, sandboxed apps load only audio unit plugins that declare themselves to be safe for use in a sandbox. With this temporary exception, the user is instead asked for permission when the app attempts to load an unsafe (or undeclared) plugin.

Entitlement key	Capability
`com.apple.security.temporary-exception.audio-unit-host`	Enables hosting of audio components that are not designated as sandbox-safe. See *Audio Components and the Application Sandbox* for details.

Global Mach Service Temporary Exception

With App Sandbox, lookup of global Mach services fails unless you configure the `mach-lookup.global.name` temporary exception entitlement. For each service that you want to enable, add the service as a string value for this entitlement key's value array.

Entitlement key	Capability
`com.apple.security.temporary-exception.mach-lookup.global-name`	Enables lookup of one or more global Mach services.

File Access Temporary Exceptions

With App Sandbox, your app has access only to its container, to its application group containers, to locations that are POSIX world-readable, and to locations in the file system that the user indicates direct intent to use, such as by interacting with an Open or Save dialog. If your app needs permanent access to other locations, you can bring additional locations into your sandbox by enabling the temporary exception entitlement keys described here.

For each path that you want to enable access to, specify the path as a string value for the appropriate entitlement key's value array. Each string must start with a slash (`/`) character—whether it represents an absolute path or a path relative to the user's home directory. If a path you provide specifies a directory rather a file, you must end the path with a slash character.

- For a `home-relative-path` temporary exception, provide a path relative to the user's home directory; that is, relative to `~`
- For an `absolute-path` temporary exception, provide an absolute path; that is, relative to `/`

Do not use a read/write entitlement when a read-only entitlement will do.

Important: Do not use either of these temporary exception entitlements (`home-relative-path` or `absolute-path`) to obtain access to a shared preference file. Instead, use a `shared-preference` temporary exception entitlement, as described in Shared Preference Domain Temporary Exceptions (page 30).

Entitlement key	Capability
`com.apple.security.temporary-exception.files.home-relative-path.read-only`	Enables read-only access to the specified files or subdirectories in the user's home directory.
`com.apple.security.temporary-exception.files.home-relative-path.read-write`	Enables read/write access to the specified files or subdirectories in the user's home directory.
`com.apple.security.temporary-exception.files.absolute-path.read-only`	Enables read-only access to the specified files or directories at specified absolute paths.

Entitlement key	Capability
com.apple.security.temporary-exception.files.absolute-path.read-write	Enables read/write access to the specified files or directories at specified absolute paths.

Shared Preference Domain Temporary Exceptions

If your app needs read-only or read/write access to a shared preference domain, use the following entitlements. Do not use a read/write entitlement when a read-only entitlement will do.

Entitlement key	Capability
com.apple.security.temporary-exception.shared-preference.read-only	Enables read-only access to the contents of the specified preference domain or domains in the user's home directory.
com.apple.security.temporary-exception.shared-preference.read-write	Enables read/write access to the contents of the specified preference domain or domains in the user's home directory.

Document Revision History

This table describes the changes to *Entitlement Key Reference*.

Date	Notes
2014-09-17	Added a description of the com.apple.security.device.audio-video-bridging key.
2013-12-16	Corrected the description of the com.apple.security.scripting-targets entitlement. See Enabling Scripting of Other Apps (page 24) for details.
2013-03-14	Clarified various sandbox-related entitlements.
2012-09-19	Added a new section for obtaining access to shared preferences, Shared Preference Domain Temporary Exceptions (page 30).
2012-07-20	Added a description of the app group entitlement key, in Adding an App to an App Group (page 22). Corrected the guidance for iCloud container identifiers in Enabling iCloud Document Storage (page 8).
2012-05-14	Updated the information on security-scoped bookmarks for OS X v10.7.4 in Enabling Security-Scoped Bookmark and URL Access (page 19).
2012-02-16	Improved the description for the com.apple.security.device.microphone entitlement key in App Sandbox Entitlement Keys (page 11). Added descriptions for new entitlement keys for OS X v10.7.3 in Enabling Security-Scoped Bookmark and URL Access (page 19).

Date	Notes
2011-09-28	New document that describes the entitlement keys for App Sandbox, iCloud, and push notifications. Some of the content in this document was previously available in *Code Signing and Application Sandboxing Guide*.

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