Framework

System Extensions

Install and manage user space code that extends the capabilities of macOS.

macOS 10.15+

Overview

Extend the capabilities of macOS by installing and managing system extensions—drivers and other low-level code—in user space rather than in the kernel. By running in user space, system extensions can't compromise the security or stability of macOS. The system grants these extensions a high level of privilege, so they can perform the kinds of tasks previously reserved for kernel extensions (KEXTs).

You use frameworks like DriverKit, Endpoint Security, and Network Extension to write your system extension, and you package the extension in your app bundle. At runtime, use the SystemExtensions framework to install or update the extension on the user's system. Once installed, an extension remains available for all users on the system. Users can disable the extension by deleting the app, which deletes the extension.

Configure the system extension and the host app

To successfully activate your extension, you must adhere to the following rules:

- The extension must match your bundle identifier, excluding file extension. For example, a DriverKit extension with bundle identifier `com.example.usbdriver` must use the filename `com.example.usbdriver.dext`. Similarly, a NetworkExtension extension with bundle identifier `com.example.networkextension` must use the filename `com.example.networkextension.systemextension`.
- You must use the same Team ID when signing the extension that you use for signing your app, unless the extension has the `com.apple.developer.system-extension.redistributable` entitlement.

- You must either distribute your app and extension through the Mac App Store, or notarize them. See Notarizing macOS software before distribution to learn more about notarization.

Topics

Essentials

Implementing drivers, system extensions, and kexts

Create drivers and system extensions to communicate with hardware and provide low-level services, and only use kernel extensions for a few tasks.

Debugging and testing system extensions

Debug your system extensions by temporarily disabling the security checks that macOS performs during the installation process.

`System Extension Entitlement`

A Boolean value that indicates whether your app has permission to activate or deactivate system extensions.

Usage descriptions

`let NSSystemExtensionUsageDescriptionKey: String`

A message that tells the user why the app is trying to install a system extension bundle.

`let OSBundleUsageDescriptionKey: String`

A message that tells the user why the app is trying to install a driver extension bundle.

Extension activation and deactivation

Installing System Extensions and Drivers

Activate system extensions and drivers to make them available to the system, and update or deactivate them as needed.

`class OSSystemExtensionManager`

A type that facilitates activation and deactivation of system extensions.

`class OSSystemExtensionRequest`

A request to activate or deactivate a system extension.

`System Extension Redistributable Entitlement`

A Boolean value that indicates whether other development teams may distribute a system extension you create.

Errors

`struct OSSystemExtensionError`

An error that describes a failed extension manager request.

`enum Code`

Error codes for system extensions.

`let OSSystemExtensionErrorDomain: String`

The error domain identifying system extension errors.

Reference

SystemExtensions Constants

Classes

`class OSSystemExtensionInfo`

`class OSSystemExtensionsWorkspace`

Protocols

`protocol OSSystemExtensionsWorkspaceObserver`