Framework

DriverKit

Develop device drivers that run in user space.

DriverKit 19.0+ | iOS 16.0+ | iPadOS 16.0+ | macOS 10.15+

Overview

The DriverKit framework defines the fundamental behaviors for device drivers in macOS and iPadOS. The C++ classes of this framework define your driver's basic structure, and provide support for handling events and allocating memory. This framework also supports appropriate types for examining the numbers, strings, and other types of data in your driver's I/O registry entry. Other frameworks, such as USBDriverKit, HIDDriverKit, NetworkingDriverKit, PCIDriverKit, SerialDriverKit, and AudioDriverKit, provide the specific behaviors you need to support different types of devices.

The drivers you build with DriverKit run in user space, rather than as kernel extensions, which improves system stability and security. You create your driver as an app extension and deliver it inside your existing app.

In macOS, use the System Extensions framework to install and upgrade your driver. In iPadOS, the system automatically discovers and upgrades drivers along with their host apps.

Note

The base DriverKit framework is available in macOS for Apple silicon and Intel-based Mac computers, and in iPadOS for devices with an M-series chip. The availability of family frameworks like USBDriverKit and AudioDriverKit varies by platform.

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.

Creating drivers for iPadOS

Bring your drivers to iPadOS by using the platform's DriverKit support.

Entitlements

Requesting Entitlements for DriverKit Development

Request the entitlement for DriverKit development, and request other entitlements your driver needs to interact with specific devices and interfaces.

`com.apple.developer.driverkit`

A Boolean value that indicates whether your extension has permission to run as a user-space driver.

`com.apple.developer.driverkit.userclient-access`

An array of strings that represent macOS driver extensions that may communicate with other DriverKit services.

`com.apple.developer.driverkit.allow-any-userclient-access`

A Boolean value that determines whether a macOS driver accepts user client connections from any application.

`Communicates with Drivers`

A Boolean value that indicates whether an iPadOS app can communicate with drivers.

`DriverKit Allow Third Party User Clients`

A Boolean value that indicates whether an iPadOS driver accepts calls from third-party user clients.

Samples

DriverKit sample code

Explore projects that demonstrate how to write macOS device drivers with the DriverKit family of frameworks.

Services

Creating a Driver Using the DriverKit SDK

Create a driver that supports proprietary features of your company's hardware devices.

Debugging and testing system extensions

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

`IOService`

The base class for managing the setup and registration of your driver.

Event management

`IODispatchQueue`

An object that manages the serial execution of blocks.

`IOInterruptDispatchSource`

A dispatch source that reports hardware-related interrupt events to your driver.

`IOTimerDispatchSource`

A dispatch source that notifies your driver at a specific time.

`IODataQueueDispatchSource`

A dispatch source that manages a shared-memory data queue.

`IODispatchSource`

The common base class for dispatch sources.

`OSAction`

An object that executes your driver's custom behavior.

Memory management

`IOBufferMemoryDescriptor`

A memory buffer allocated in the caller's address space.

`IOMemoryDescriptor`

The base class for describing a location in memory.

`IOMemoryMap`

A reference to an existing block of memory in the current process or in a different process.

Memory Utilities

Allocate and deallocate memory and manage memory pointers in different address spaces.

Registry data types

`OSArray`

A container for an ordered, random-access collection of objects.

`OSDictionary`

A container for a collection with elements that are key-value pairs.

`OSBoolean`

A container for a true or false value.

`OSData`

A container for untyped data.

`OSNumber`

A container for an integer value.

`OSString`

A container for managing an array of characters.

`OSSerialization`

A container for one or more objects, serialized in a binary data format that is suitable for messaging.

`OSCollection`

The base class for DriverKit collection objects.

`OSContainer`

The base class for DriverKit data objects.

`OSObject`

The base class for DriverKit objects

`OSSymbol`

A container for managing an array of characters.

`IOFixed`

A fixed-point number.

External drivers

`IOUserClient`

A connection to another service that the system manages.

`IOUserServer`

A system-managed service.

`com.apple.developer.driverkit.userclient-access`

An array of strings that represent macOS driver extensions that may communicate with other DriverKit services.

{} Communicating between a DriverKit extension and a client app

Send and receive different kinds of data securely by validating inputs and asynchronously by storing and using a callback.

Runtime support

`OSDynamicCast`

Casts an object safely to the specified type, if possible.

`OSRequiredCast`

Casts the object to the specified type, stopping the process if the object isn't of the correct type.

`IMPL`

Tells the system that the superclass implementation of this method runs in the kernel.

`TYPE`

Annotates a method declaration to indicate that it conforms to an existing method signature.

`QUEUENAME`

Tells the system to execute a method on the dispatch queue with the specified name.

`SUPERDISPATCH`

Tells the system to execute the superclass' implementation of the current method in the kernel.

`IIG_KERNEL`

Tells the system that the class or method runs inside the kernel.

`LOCAL`

Tells the system that the method runs locally in the driver extension’s process space.

`LOCALONLY`

Tells the system that the class or method runs locally in the driver extension’s process space.

Error Codes

Determine the reason an operation fails.

C++ Runtime Support

Examine low-level types that DriverKit uses to support kernel-level operations.

Classes

`IOHistogramReporter`

`IOReportLegend`

`IOReporter`

`IOServiceStateNotificationDispatchSource`

`IOSimpleReporter`

`IOStateReporter`

`OSBundle`

`OSMappedFile`

Reference

DriverKit Structures

DriverKit Enumerations

DriverKit Constants

DriverKit Functions

DriverKit Data Types

DriverKit Namespaces

Macros

Macros

`kIOPropertyHashTypeKey`

`kIOPropertySHA3256Key`

`kIOPropertySHA3384Key`

`kIOPropertySHA3512Key`

Enumeration Cases

`kIOServicePMAssertionCPUBit`

kIOServicePMAssertionCPUBit When set, PM kernel will prefer to leave the CPU and core hardware running in “Dark Wake” state, instead of sleeping.

`kIOServicePMAssertionForceFullWakeupBit`

kIOServicePMAssertionForceFullWakeupBit When set, the system will immediately do a full wakeup after going to sleep.

`kIOServicePowerCapabilityLPW`

`kSCSICmd_ATA_PASS_THROUGH`

`kSCSICmd_ATA_PASS_THROUGH_EXT`