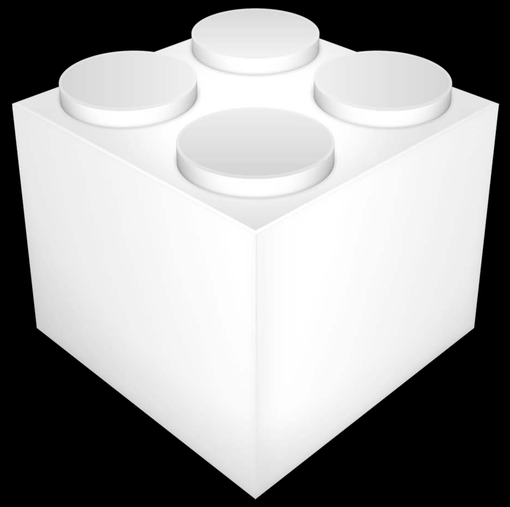
#WWDC19

System Extensions and DriverKit

Modernizing kernel extensions and device drivers

Joe Auricchio, Darwin Runtime
Simon Douglas, Core Kernel
Scott Deandrea, Core I/O

kexts



(Kernel Extensions)

The power to extend the operating system

Part of what makes the Mac, the Mac

Part of what makes the Mac, the Mac

Kexts Have Some Problems

Kexts Have Some Problems

Difficult to develop and debug

Kexts Have Some Problems

Difficult to develop and debug

Risk to data security and privacy

Kexts Have Some Problems

Difficult to develop and debug

Risk to data security and privacy

Risk to reliability

It’s time for an upgrade

NEW

System Extensions and DriverKit

Extend the operating system

Extend the operating system

More reliable

Extend the operating system

More reliable

More secure

Extend the operating system

More reliable

More secure

Easier to develop

Agenda

Agenda

Introducing System Extensions and DriverKit

Agenda

Introducing System Extensions and DriverKit

How they avoid the problems of Kernel Extensions

Agenda

Introducing System Extensions and DriverKit

How they avoid the problems of Kernel Extensions

Building Driver Extensions with DriverKit

Agenda

Introducing System Extensions and DriverKit

How they avoid the problems of Kernel Extensions

Building Driver Extensions with DriverKit

USB Driver Demo

Agenda

Introducing System Extensions and DriverKit

How they avoid the problems of Kernel Extensions

Building Driver Extensions with DriverKit

USB Driver Demo

System Extensions in your apps

A System Extension Is

NEW

A System Extension Is

NEW

Part of your app

A System Extension Is

NEW

Part of your app

That extends the functionality of the operating system

A System Extension Is

NEW

Part of your app

That extends the functionality of the operating system

In similar ways to a kernel extension

A System Extension Is

NEW

Part of your app

That extends the functionality of
the operating system

In similar ways to a kernel extension

But running in userspace, outside the kernel

Kinds of System Extensions

Network Extensions

Endpoint Security Extensions

Driver Extensions

Network Extensions

Network Extensions

Replacement for Network Kernel Extensions

Network Extensions

Replacement for Network Kernel Extensions

Supported capabilities

- Content filter
- DNS proxy
- VPN client

Network Extensions

Replacement for Network Kernel Extensions

Supported capabilities

- Content filter
- DNS proxy
- VPN client

Network Extensions for the Modern Mac

Friday, 9:00

Endpoint Security Extensions

Endpoint Security Extensions

Replacement for `kauth` event monitoring

Endpoint Security Extensions

Replacement for `kauth` event monitoring

Example types of apps

- Endpoint Detection and Response
- Anti-virus
- Data Loss Prevention

Endpoint Security Extensions

Replacement for `kauth` event monitoring

Example types of apps

- Endpoint Detection and Response
- Anti-virus
- Data Loss Prevention

Security Lab 1	Tuesday, 10:00
Security Lab 2	Thursday, 2:00

Driver Extensions

Driver Extensions

Replacement for IOKit device drivers

Driver Extensions

Replacement for IOKit device drivers

Control hardware devices and vend services to the system

- USB
- Serial
- NIC (Network Interface Controller)
- HID (Human Interface Device)

Driver Extensions

Replacement for IOKit device drivers

Control hardware devices and vend services to the system

- USB
- Serial
- NIC (Network Interface Controller)
- HID (Human Interface Device)

Built using DriverKit

NEW

DriverKit Is...

NEW

DriverKit Is...

A new SDK with all-new frameworks

NEW

DriverKit Is...

based on IOKit, but updated and modernized

NEW

DriverKit Is...

for building device drivers in userspace, outside the kernel

How Do System Extensions Avoid the Problems of Kernel Extensions?

In userspace, outside the kernel

The Kernel Is an Unforgiving Environment

The Kernel Is an Unforgiving Environment

The kernel must never stop running, must never wait, must never crash

The Kernel Is an Unforgiving Environment

The kernel must never stop running, must never wait, must never crash

Code in the kernel has to be

- Fast
- Predictable
- Frugal with resources like memory
- Bug-free

The Kernel Is an Unforgiving Environment

The kernel must never stop running, must never wait, must never crash

Code in the kernel has to be

- Fast
- Predictable
- Frugal with resources like memory
- Bug-free

Difficult to write and debug kernel code

In userspace, outside the kernel

Comfortable, modern programming environment

Kexts vs. System Extensions

Easier development

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory
allocation, synchronization, latency

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Only C/C++ supported

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Only C/C++ supported

No such restrictions

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Only C/C++ supported

No such restrictions

Use any framework in the macOS SDK

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Only C/C++ supported

No such restrictions

Use any framework in the macOS SDK

Use any language, including Swift

Kexts vs. System Extensions

Easier development

Restrictions on dynamic memory allocation, synchronization, latency

Cannot use system frameworks

Only C/C++ supported

Exception: Driver Extensions

Direct control of hardware still imposes some restrictions

DriverKit frameworks in tailored runtime

DriverKit API is C++17

Kexts vs. System Extensions

Easier debugging

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

- Can’t evaluate expressions

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

- Can’t evaluate expressions

Kernel and other apps keep running if system extension stops

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

- Can’t evaluate expressions

Kernel and other apps keep running if system extension stops

No need to restart if extension crashes

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

- Can’t evaluate expressions

Kernel and other apps keep running if system extension stops

No need to restart if extension crashes

Build, test, debug on one machine

Kexts vs. System Extensions

Easier debugging

Debugging the kernel halts everything

Two-machine debugging

- Special debug cables or LAN setup

Slow build-test-debug cycle

Limited debugger support

- Can’t evaluate expressions

Kernel and other apps keep running if system extension stops

No need to restart if extension crashes

Build, test, debug on one machine

Full debugger support

Kexts vs. System Extensions

Security, privacy, reliability

Code Signing

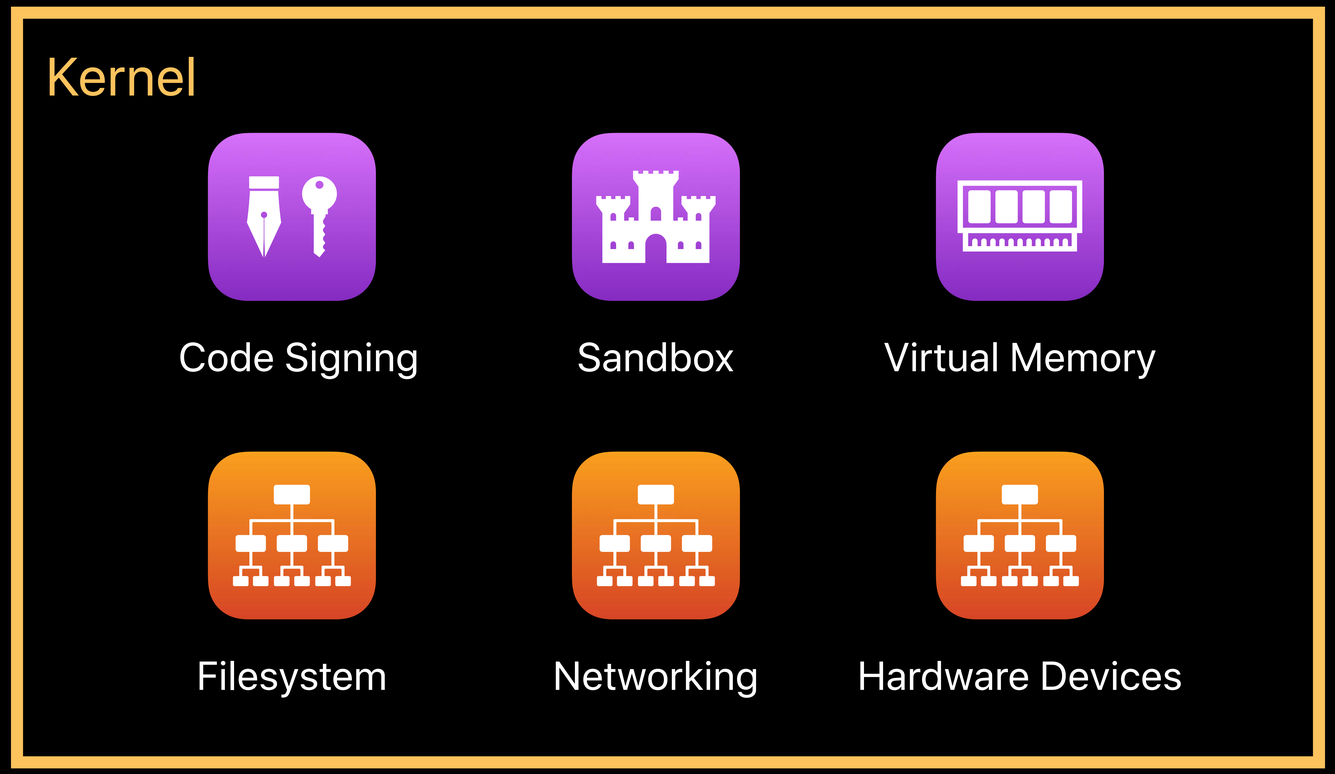
Sandbox

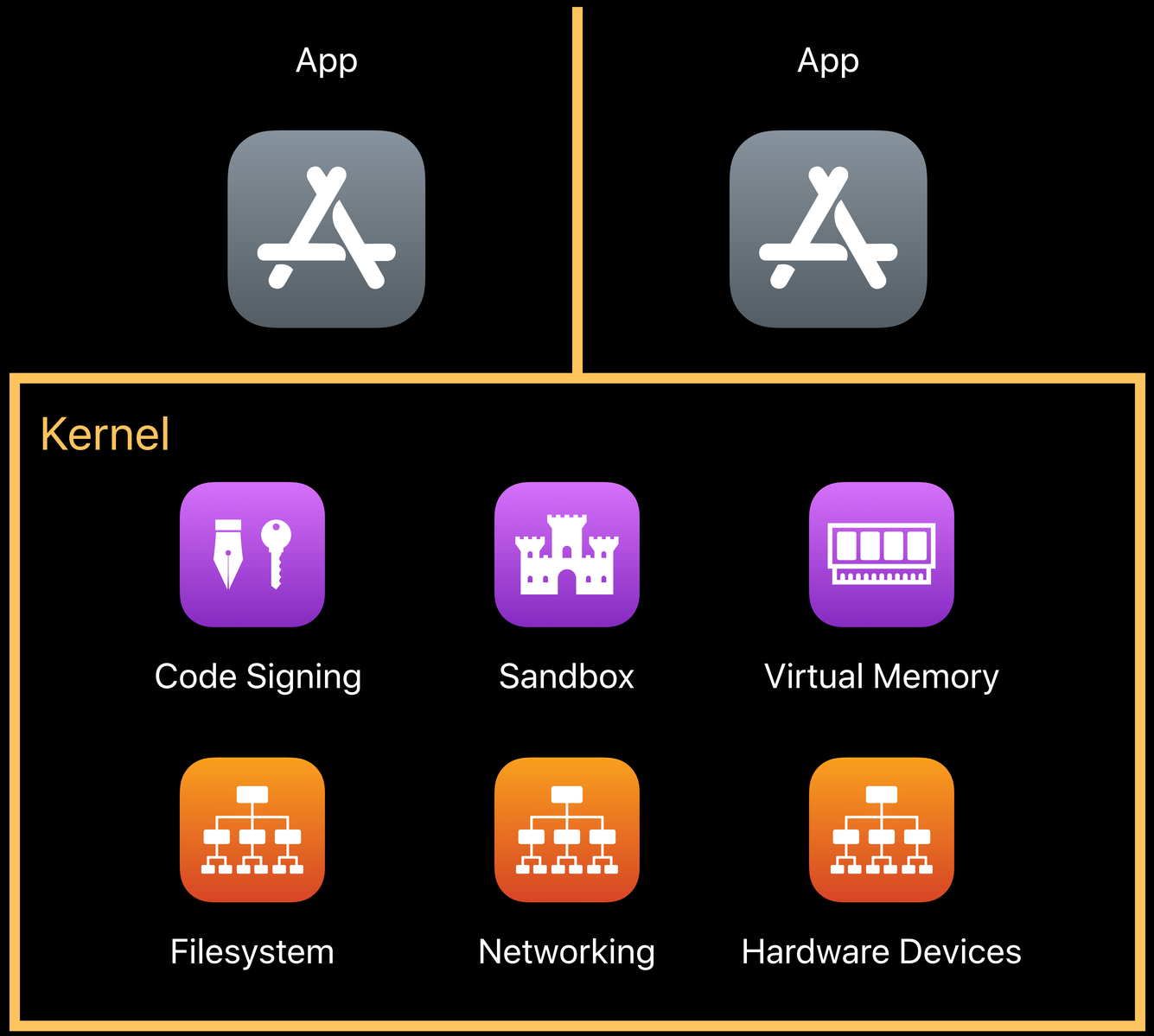
Virtual Memory

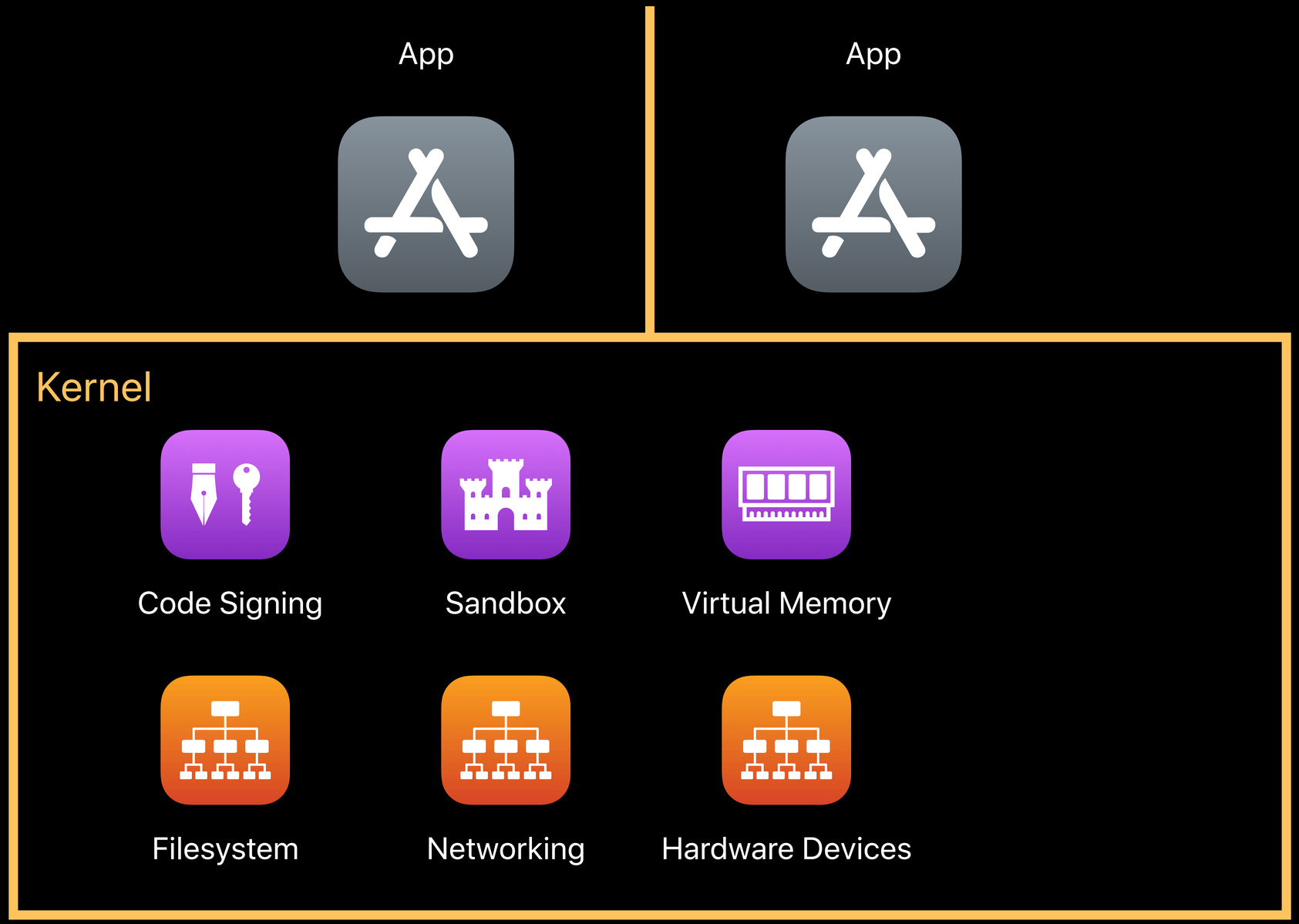
Filesystem

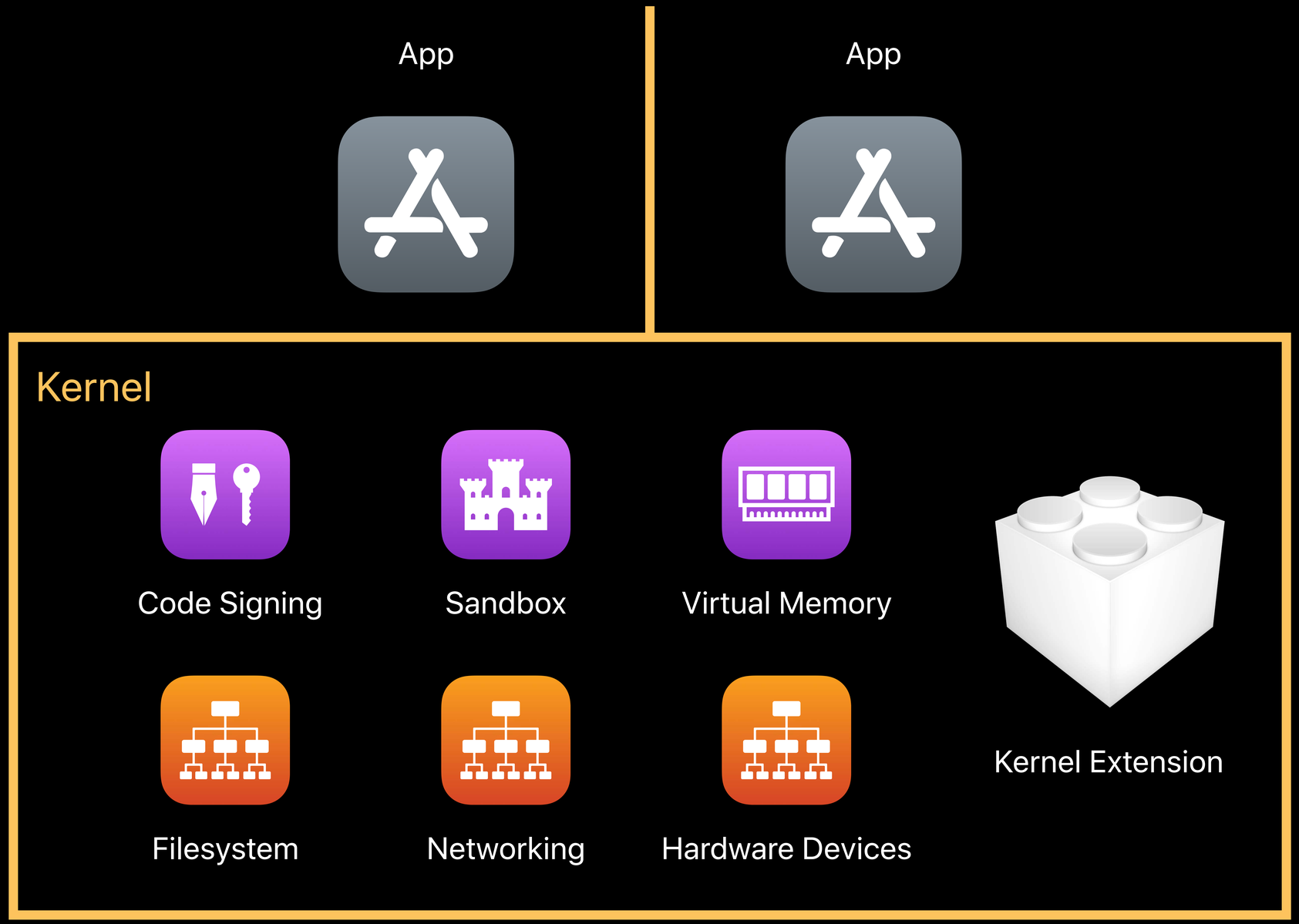
Networking

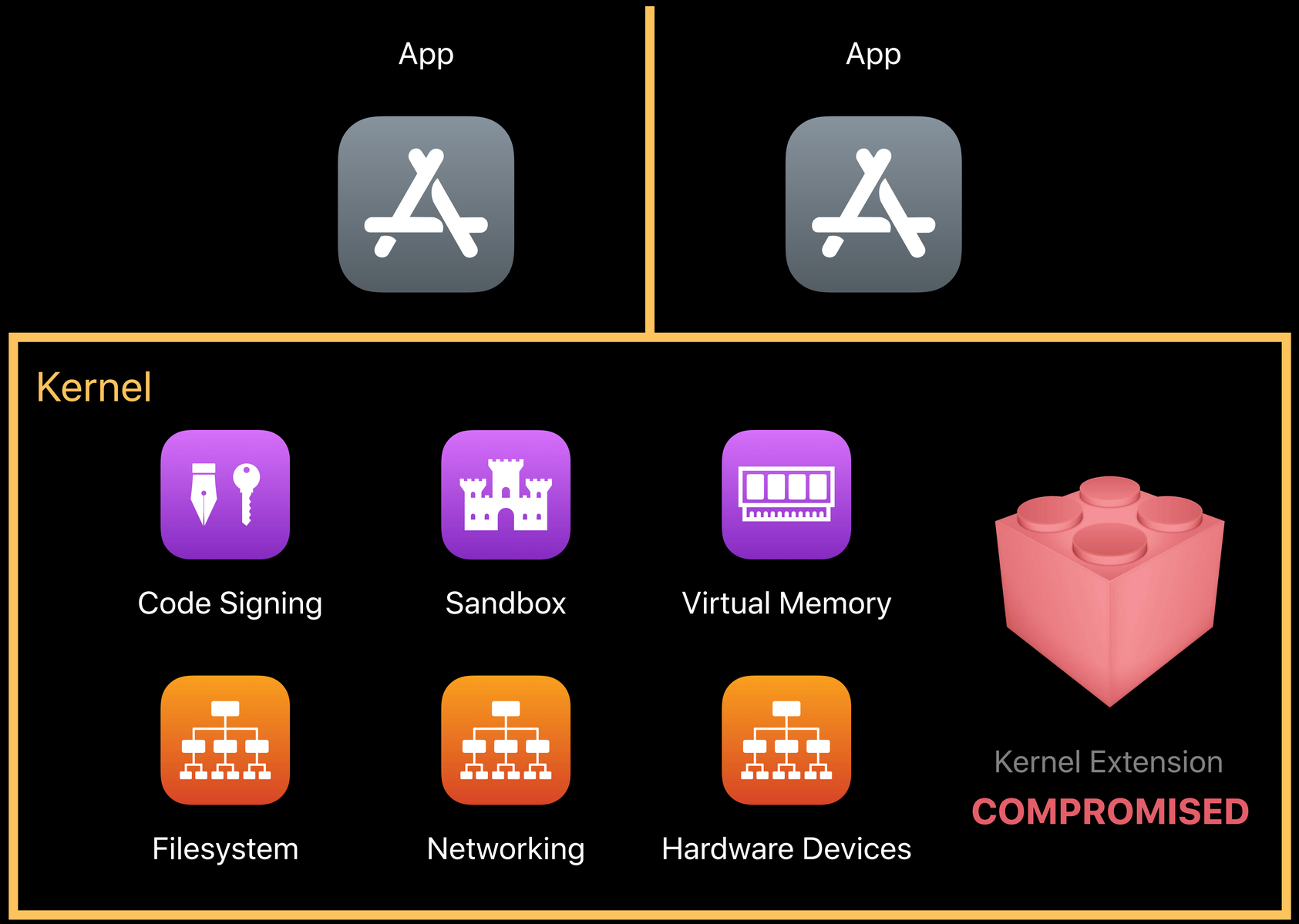
Hardware Devices

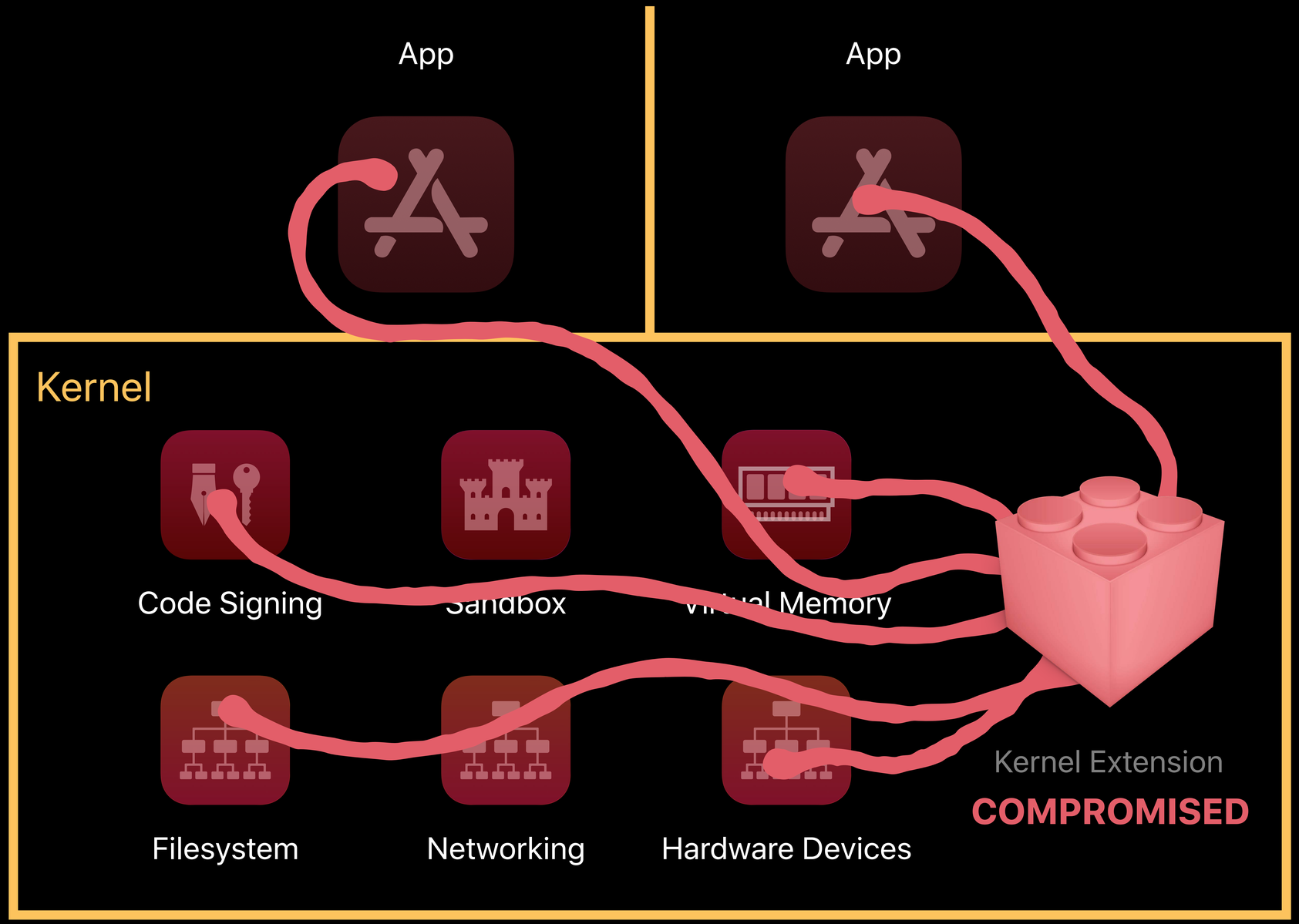

Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices

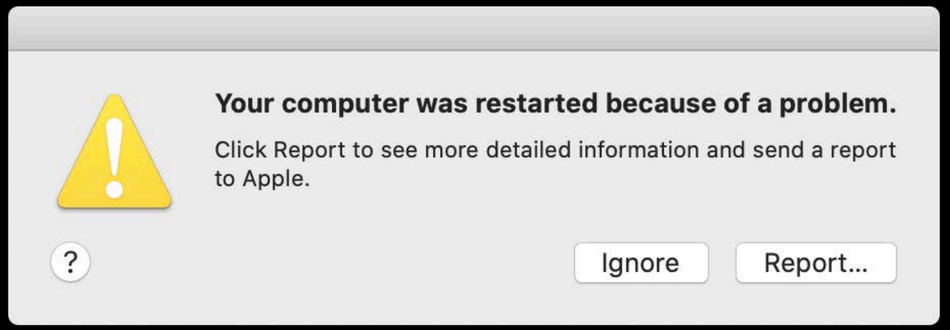

App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices

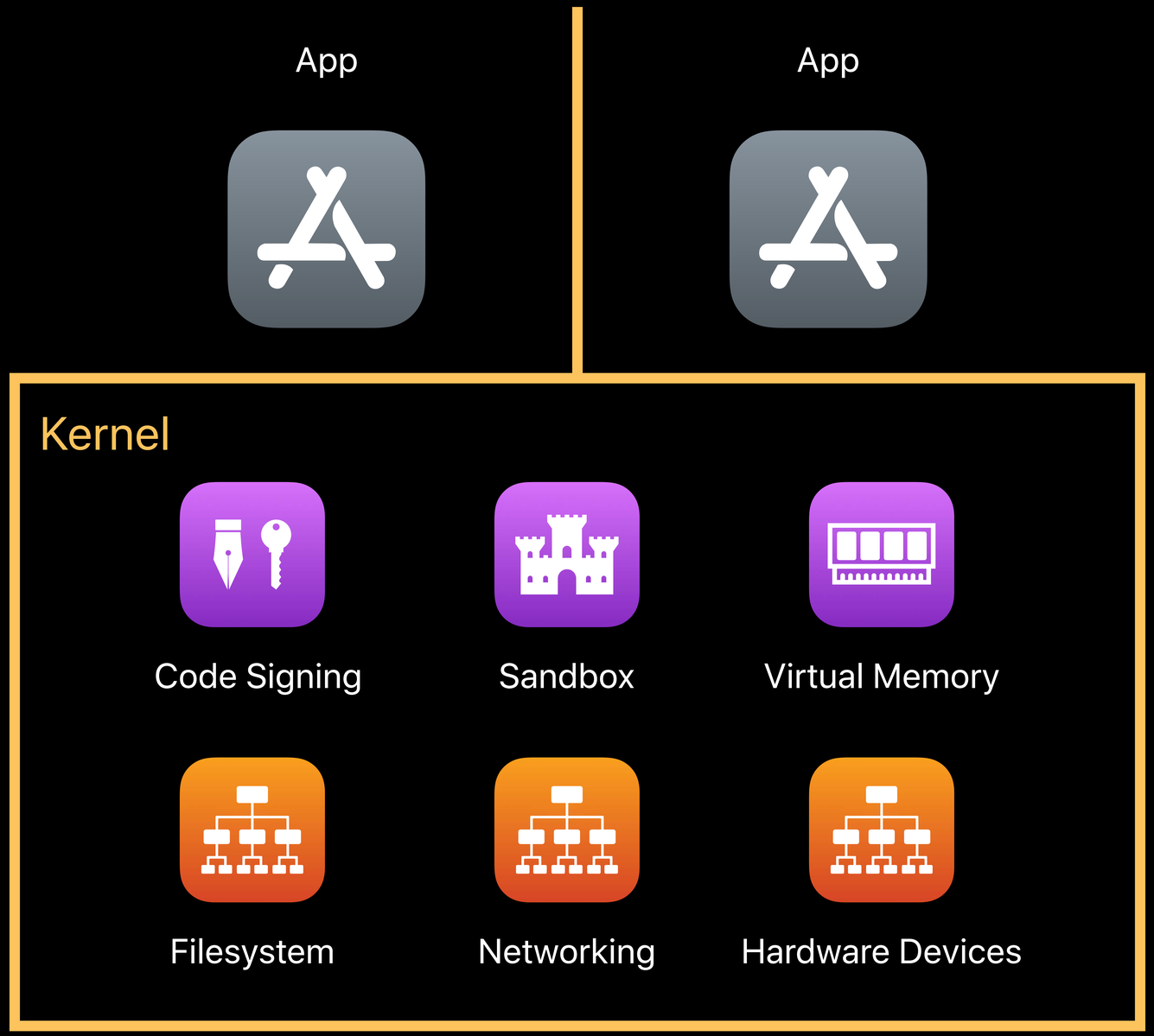

App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices

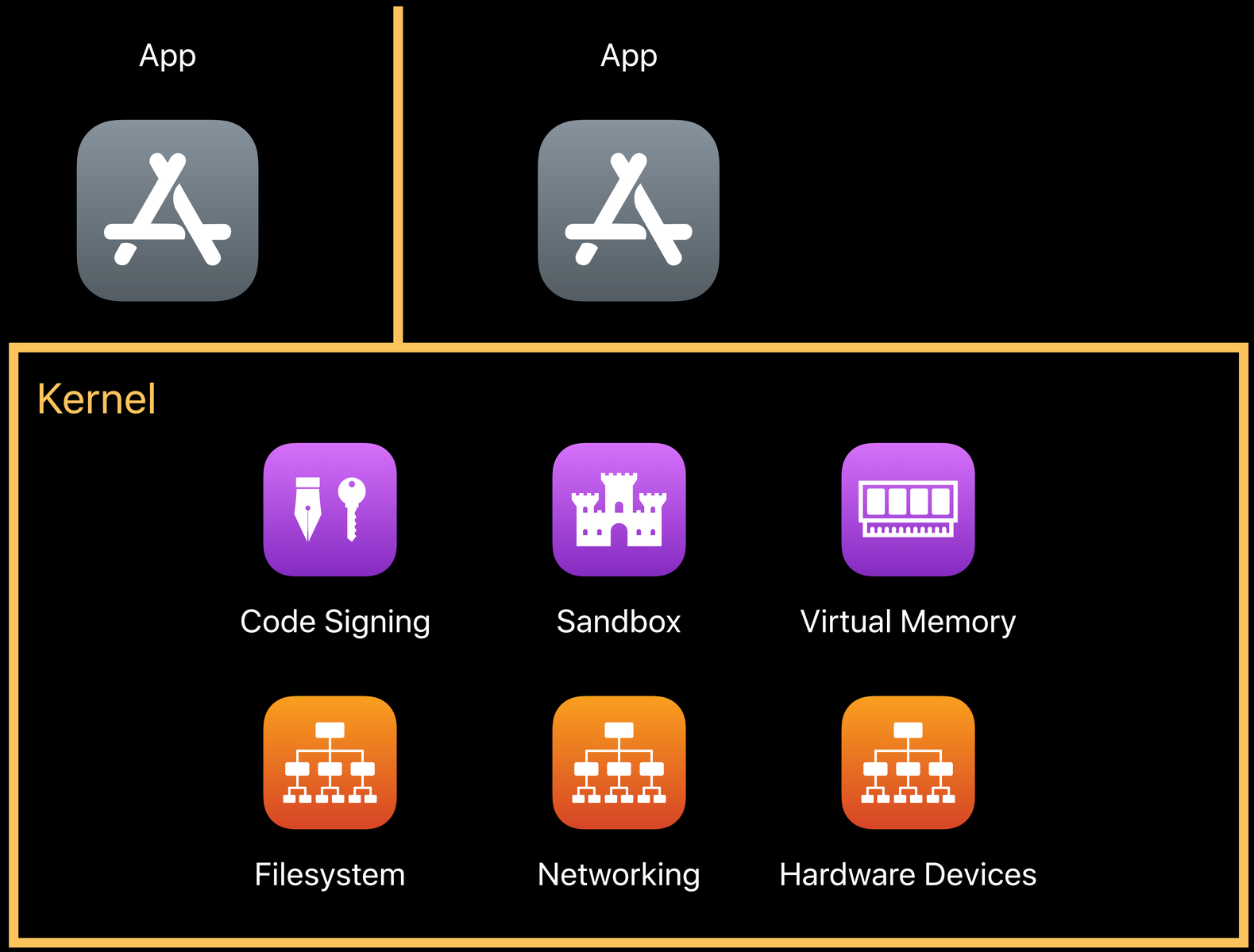

App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Kernel Extension
Filesystem
Networking
Hardware Devices

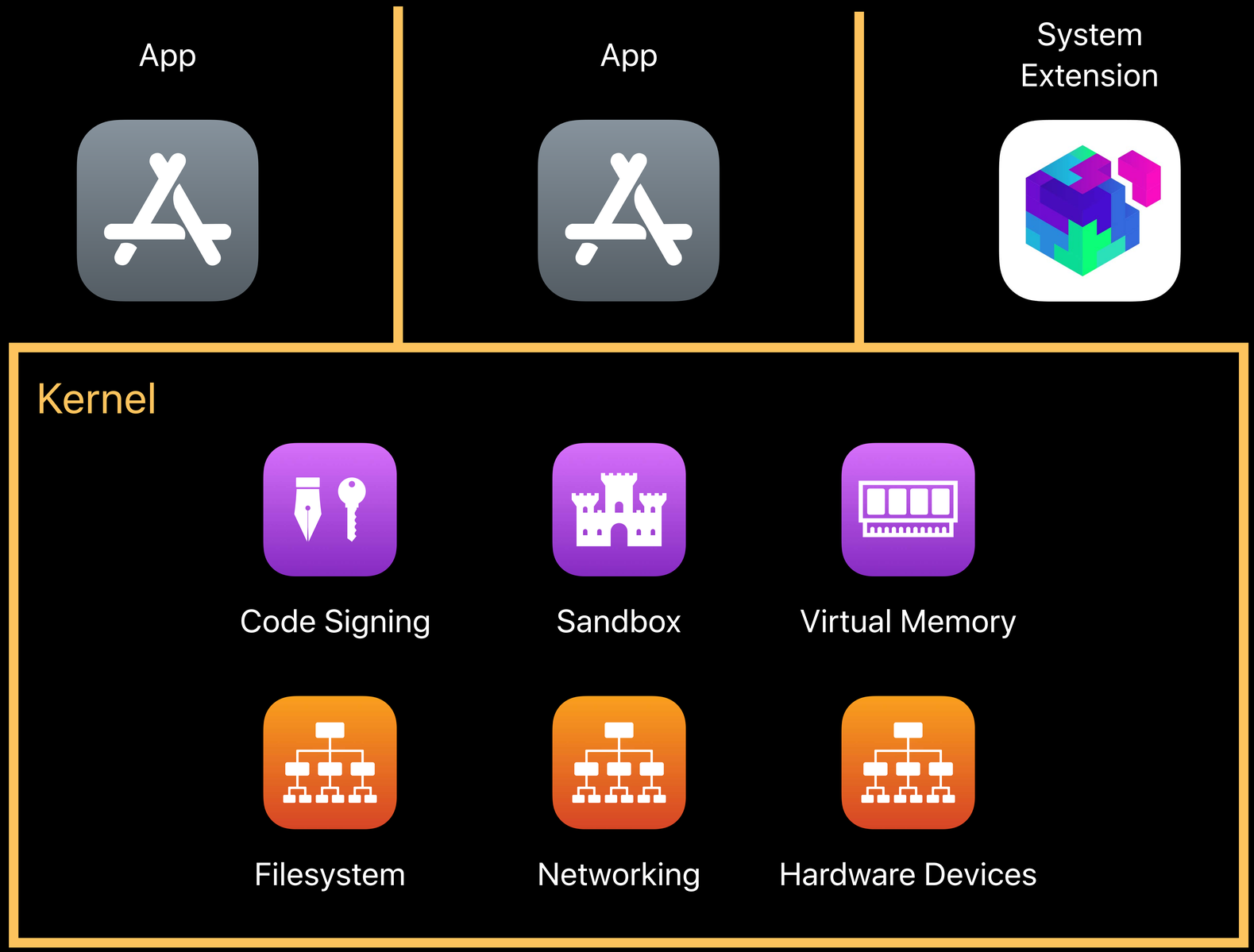

App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices
Kernel Extension
COMPROMISED

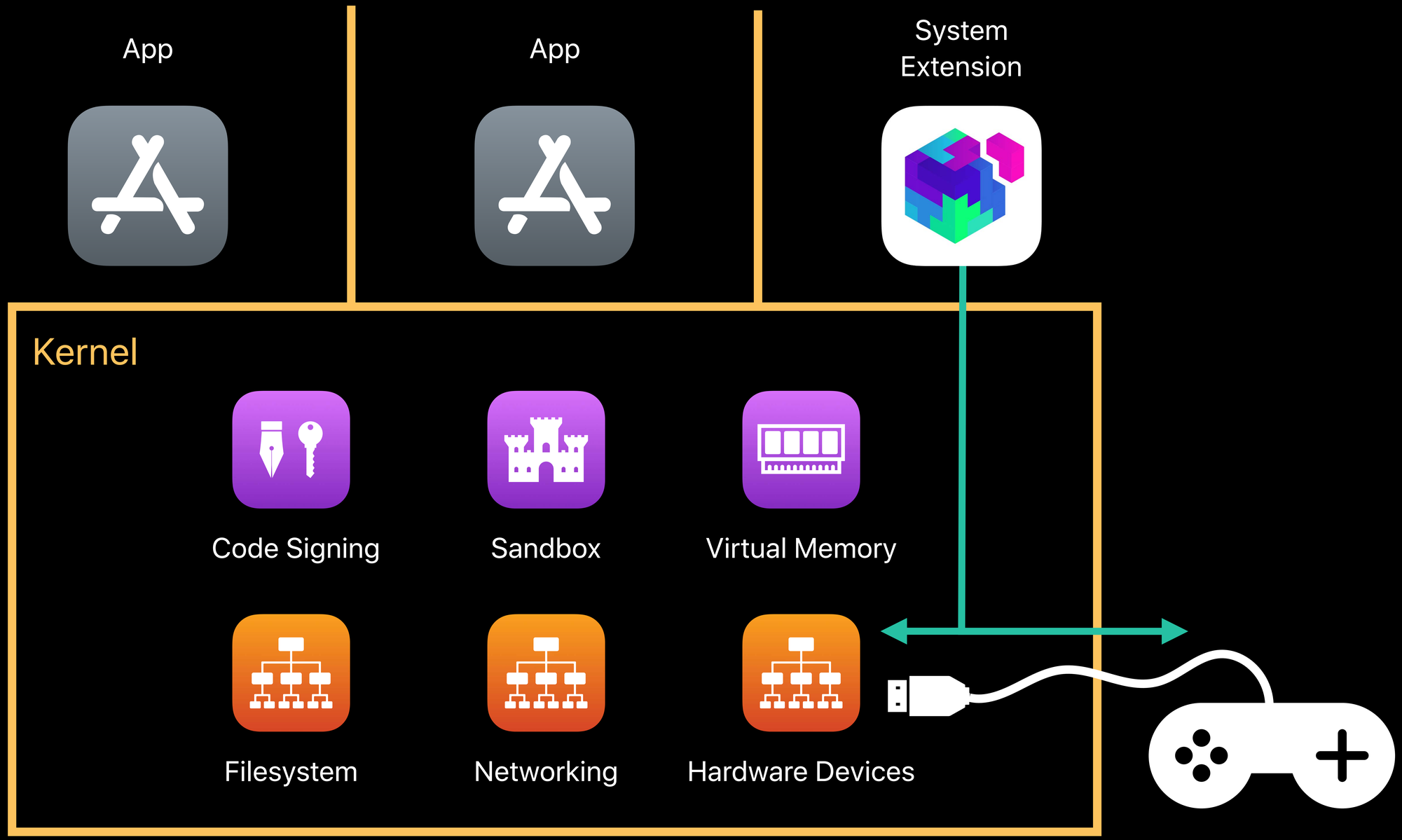

App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices
Kernel Extension
COMPROMISED

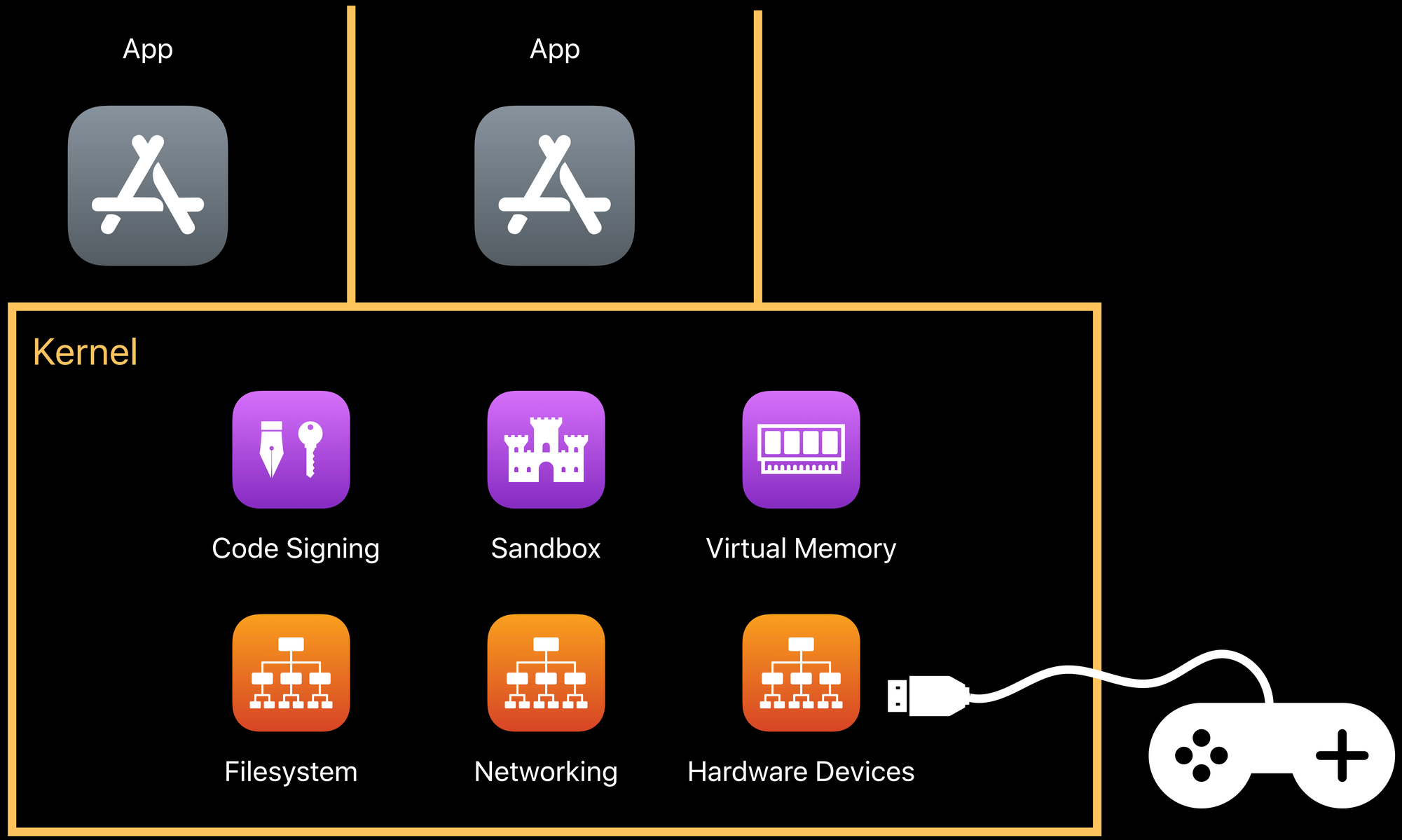

Your computer was restarted because of a problem.
Click Report to see more detailed information and send a report to Apple.
?
Ignore
Report...


App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices


App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices


App
App
System Extension
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices


App
App
System Extension
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices


App
App
Kernel
Code Signing
Sandbox
Virtual Memory
Filesystem
Networking
Hardware Devices

System Extensions are a big step forward for the Mac platform

Deprecating Kernel Extensions

Deprecating Kernel Extensions

macOS 10.15 will be the last release to fully support kexts without compromises

Deprecating Kernel Extensions

macOS 10.15 will be the last release to fully support kexts without compromises

For the capabilities and device families supported by System Extensions, using a Kernel Extension to perform the same function is deprecated

Deprecating Kernel Extensions

macOS 10.15 will be the last release to fully support kexts without compromises

For the capabilities and device families supported by System Extensions, using a Kernel Extension to perform the same function is deprecated

In a future release of macOS, Kernel Extensions of these kinds will not load

Deprecating Kernel Extensions — In the Future

Deprecating Kernel Extensions — In the Future

More kinds of System Extensions will be added

Deprecating Kernel Extensions — In the Future

More kinds of System Extensions will be added

More DriverKit device families will be added

Deprecating Kernel Extensions — In the Future

More kinds of System Extensions will be added

More DriverKit device families will be added

In turn, Kernel Extensions of those kinds will also be deprecated

System Extensions

System Extensions

Avoid the difficulties and dangers of kernel programming by running in userspace

System Extensions

Avoid the difficulties and dangers of kernel programming by running in userspace

Easier to develop and debug

System Extensions

Avoid the difficulties and dangers of kernel programming by running in userspace

Easier to develop and debug

Protect data security, privacy, reliability

Driver Extensions

Using DriverKit to build user space drivers

Simon Douglas, Core Kernel

Driver Extensions

Driver Extensions

System Extensions that control hardware

Driver Extensions

System Extensions that control hardware

Provide a service available to the entire system

Driver Extensions

System Extensions that control hardware

Provide a service available to the entire system

Driver Extension is a “dext”

Driver Extensions

Driver Extensions

Lifecycle

Driver Extensions

Lifecycle

Building

Driver Extensions

Lifecycle

Building

Security

Driver Extensions

Lifecycle

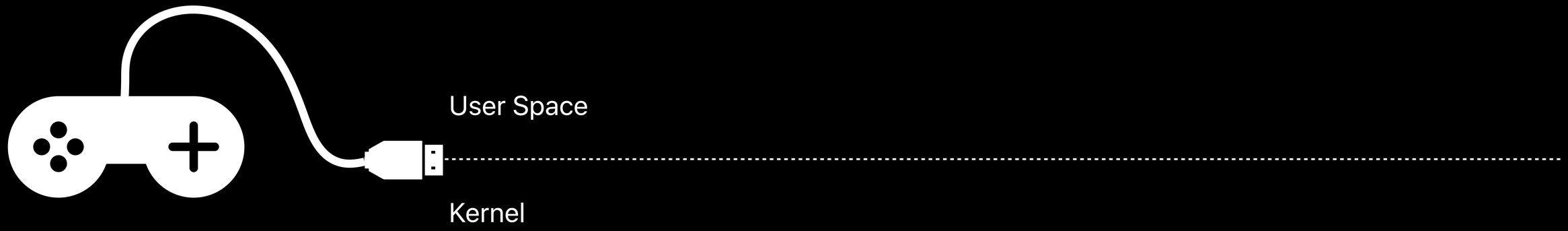
Building

Security

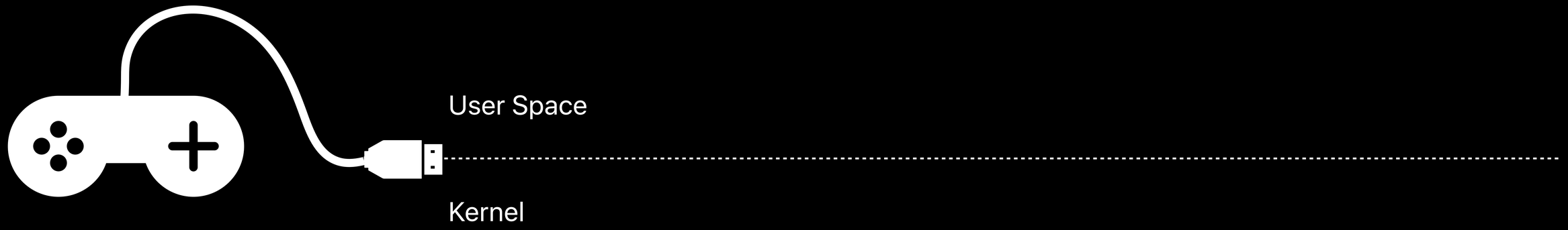
Compatibility

Lifecycle of a Dext

Lifecycle of a Dext



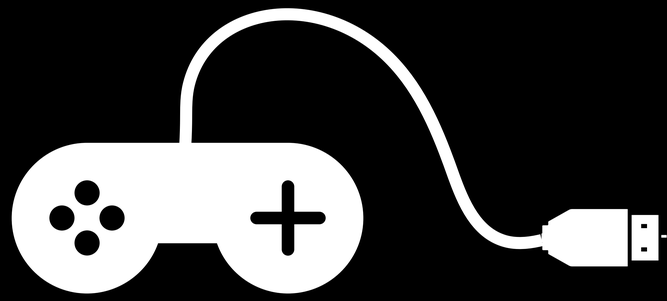
Lifecycle of a Dext



IOKit Matching

USB Device

Lifecycle of a Dext



User Space

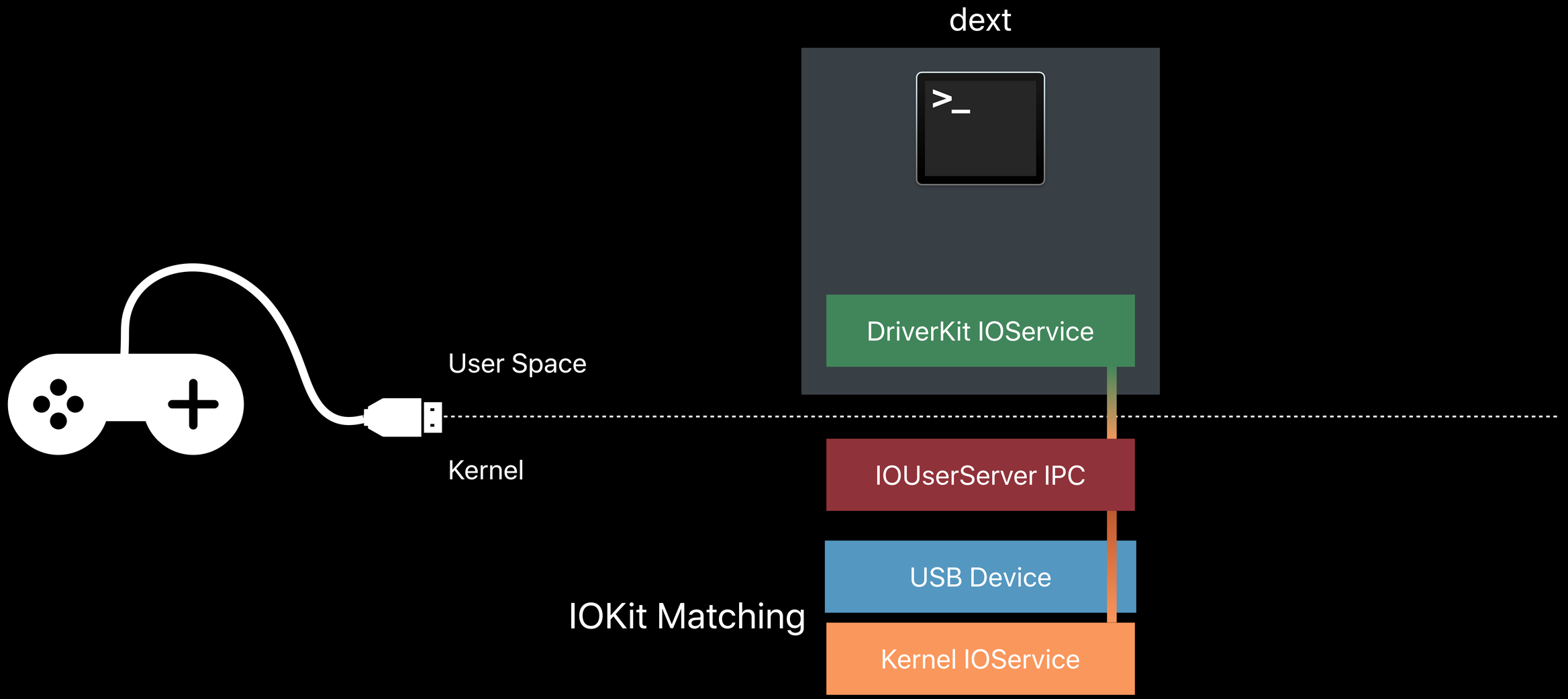
Kernel

IOKit Matching

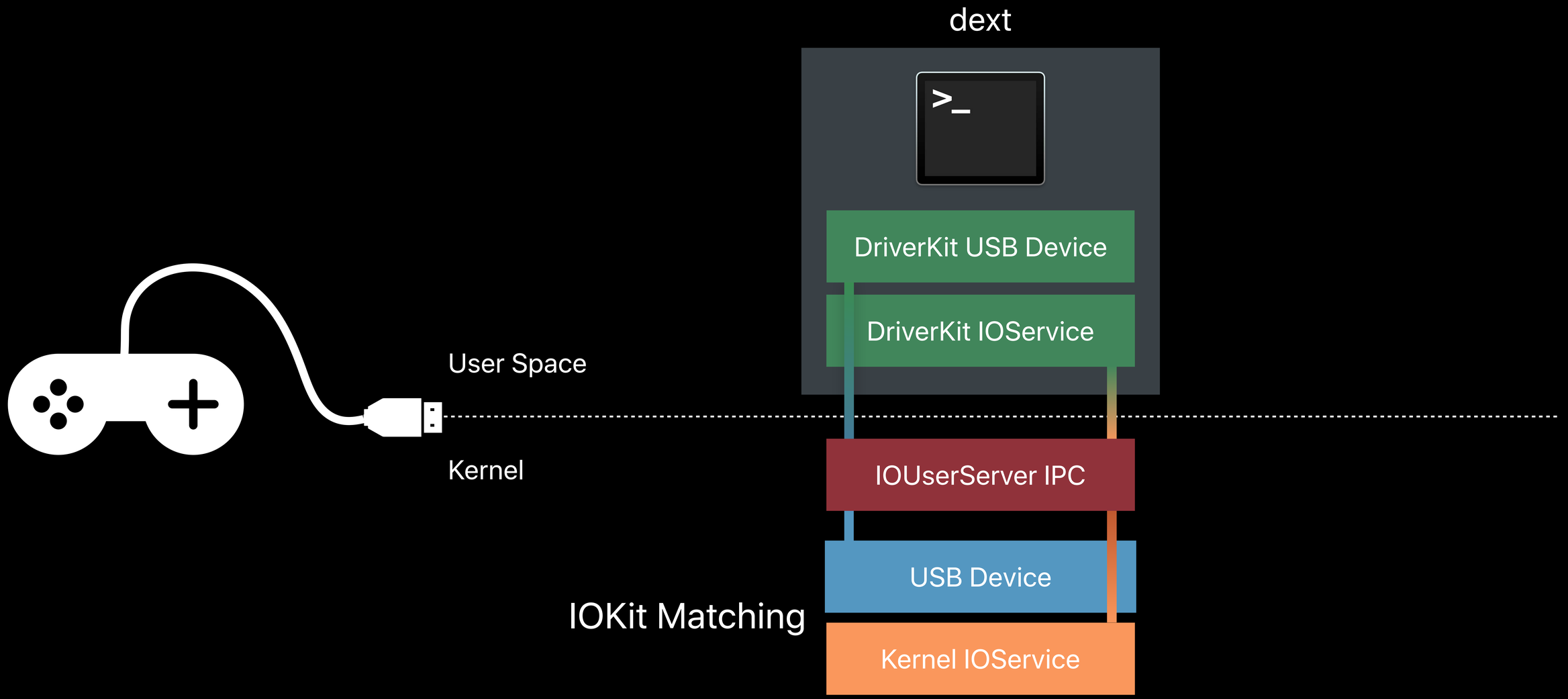
USB Device

Kernel IOService

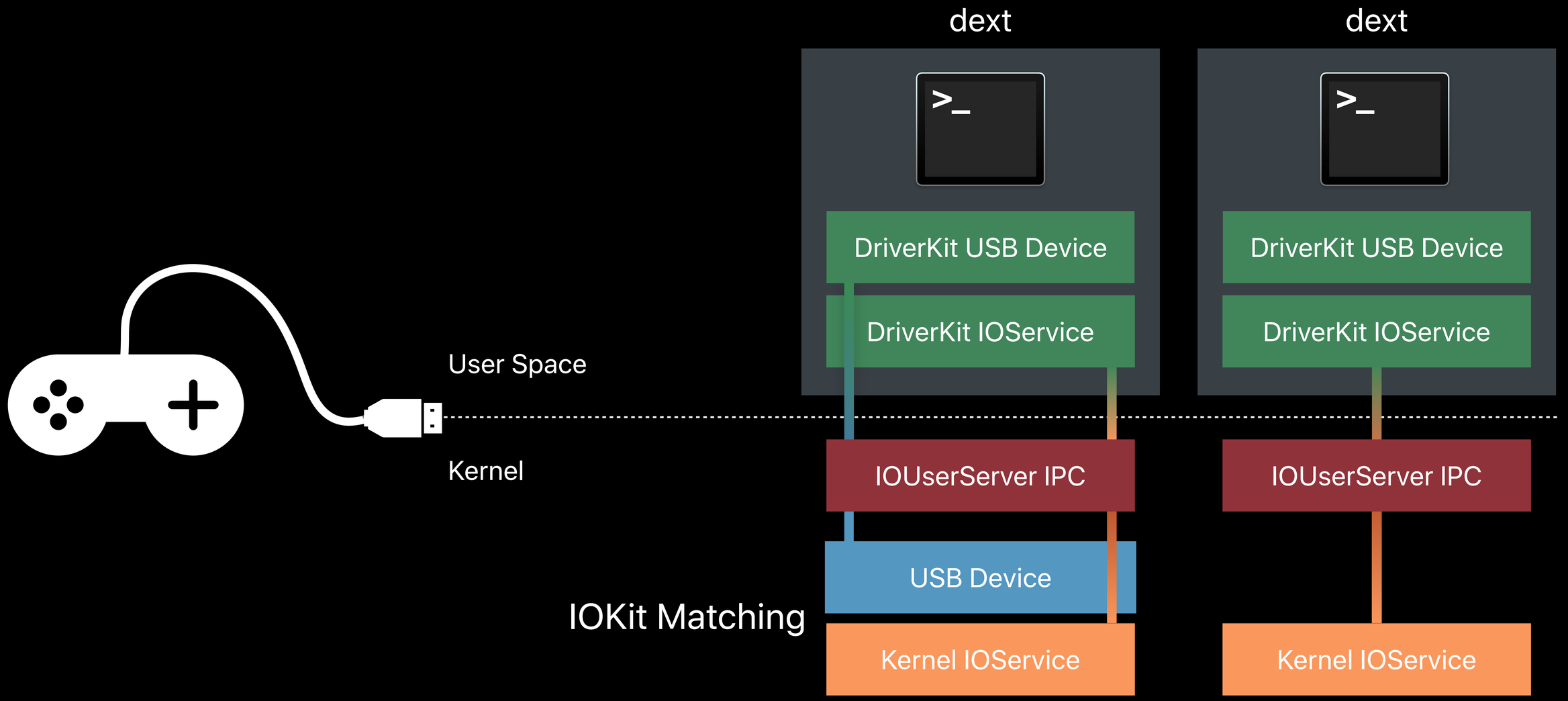
Lifecycle of a Dext



Lifecycle of a Dext



Lifecycle of a Dext



```
// Driver Extensions on macOS

~ % ps -eo user,command | grep ECM
_driverkit       /System/Library/DriverExtensions/AppleUserECM.dext/AppleUserECM
```

```
// Driver Extensions on macOS

~ % ps -eo user,command | grep ECM
_driverkit       /System/Library/DriverExtensions/AppleUserECM.dext/AppleUserECM

~ % ioreg | grep AppleUserECM
    +-o AppleUserECM  <class IOUserNetworkEthernet, id 0x1000003bc, registered, matched…
```

```
// Driver Extensions on macOS

~ % ps -eo user,command | grep ECM
_driverkit       /System/Library/DriverExtensions/AppleUserECM.dext/AppleUserECM

~ % ioreg | grep AppleUserECM
    +-o AppleUserECM  <class IOUserNetworkEthernet, id 0x1000003bc, registered, matched…

~ % ioreg | grep IOUserServer
…
    +-o IOUserServer(com.apple.driverkit.AppleUserUSBHostHIDDevice0-0x100000962) …
    +-o IOUserServer(com.apple.driverkit.AppleUserUSBHostHIDDevice0-0x100000960) …
    +-o IOUserServer(com.apple.DriverKit.AppleUSBFTDI-0x1000009a5) …
    +-o IOUserServer(com.apple.driverkit.AppleUserHIDEventDriver-0x1000009cc) …
    +-o IOUserServer(com.apple.DriverKit.AppleUserECM-0x10002db50) …
```

Building with DriverKit SDK

DriverKit SDK

DriverKit SDK

Extension of IOKit APIs

DriverKit SDK

Extension of IOKit APIs

Limited API surface

DriverKit SDK

Extension of IOKit APIs

Limited API surface

No direct access to file system, networking, IPC

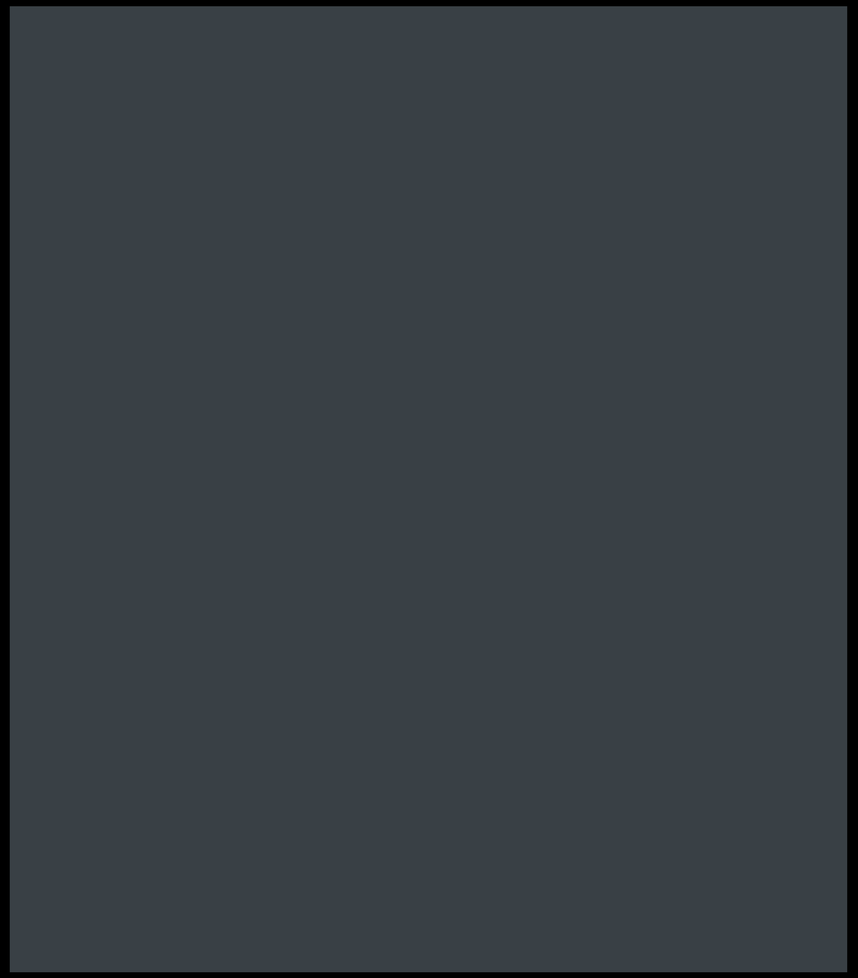
DriverKit SDK

Classes

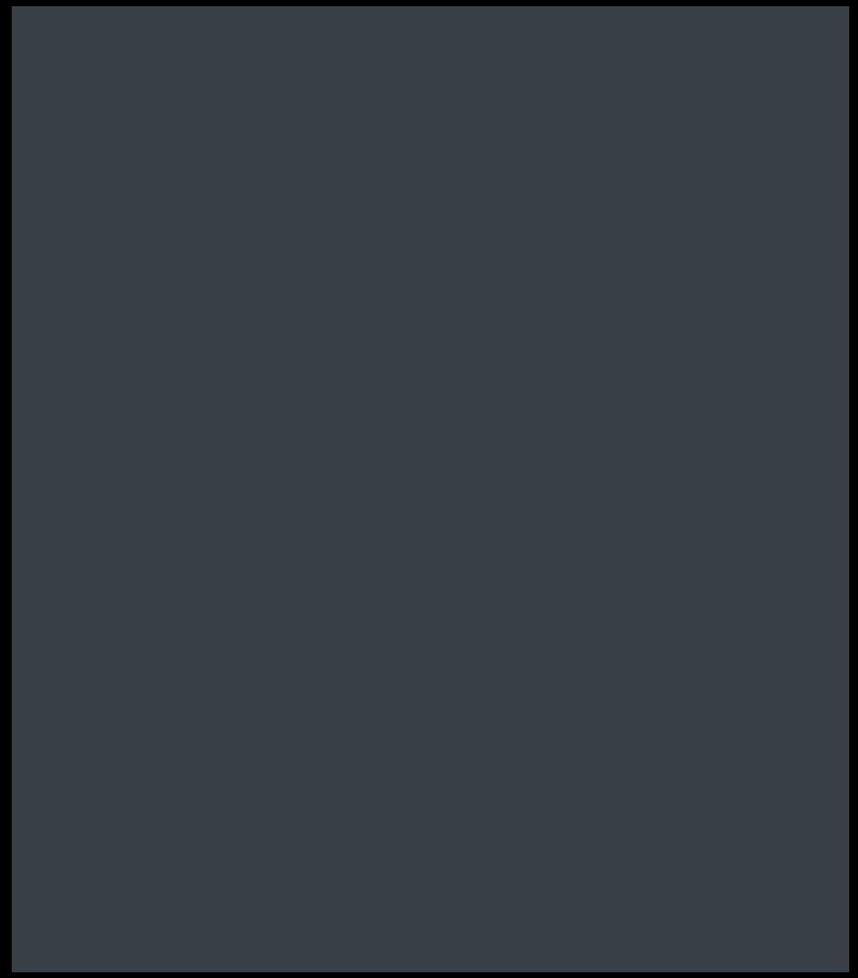
DriverKit SDK

Classes

IOKit Classes



DriverKit SDK



DriverKit SDK

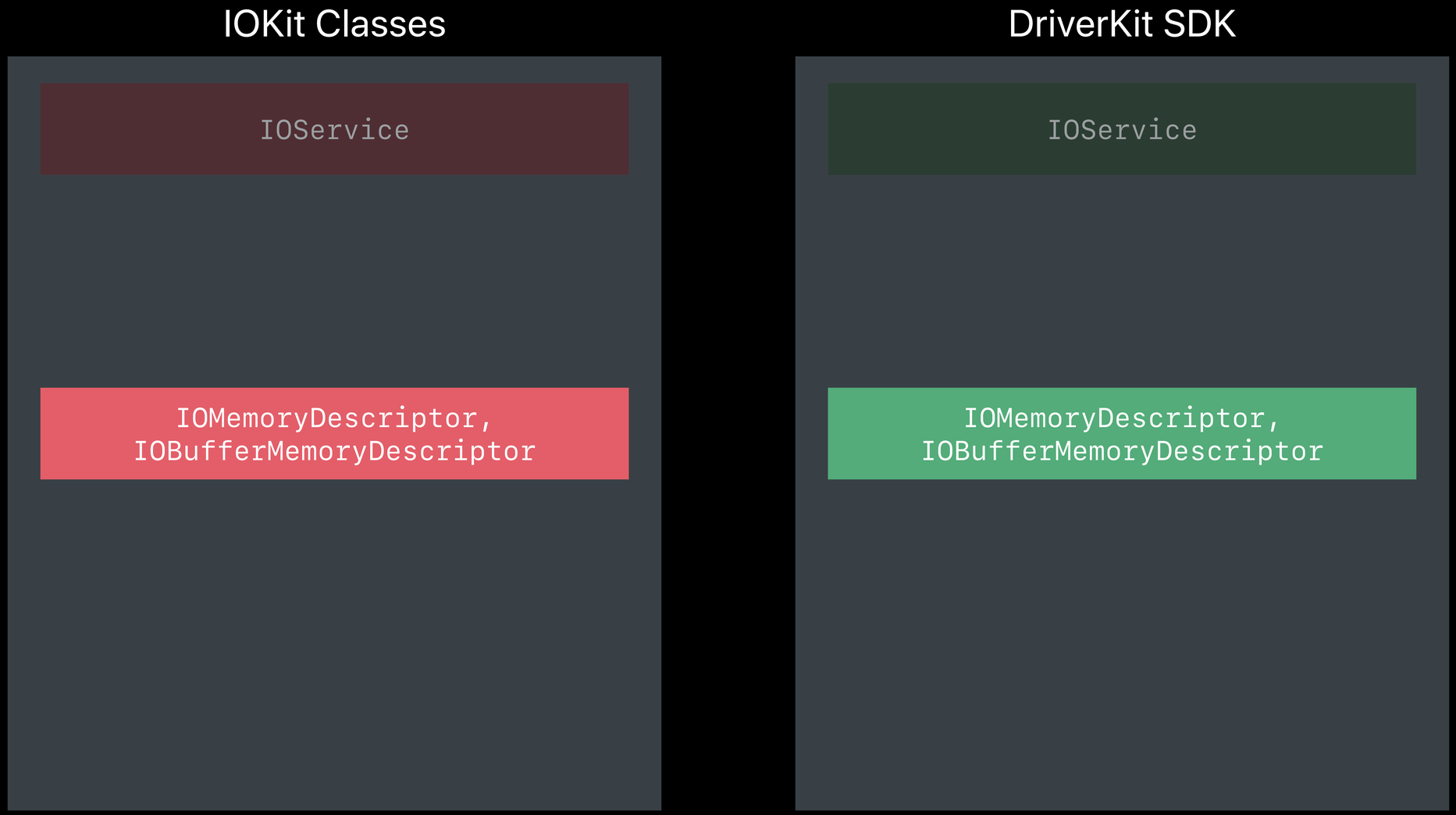
Classes

DriverKit SDK

Classes

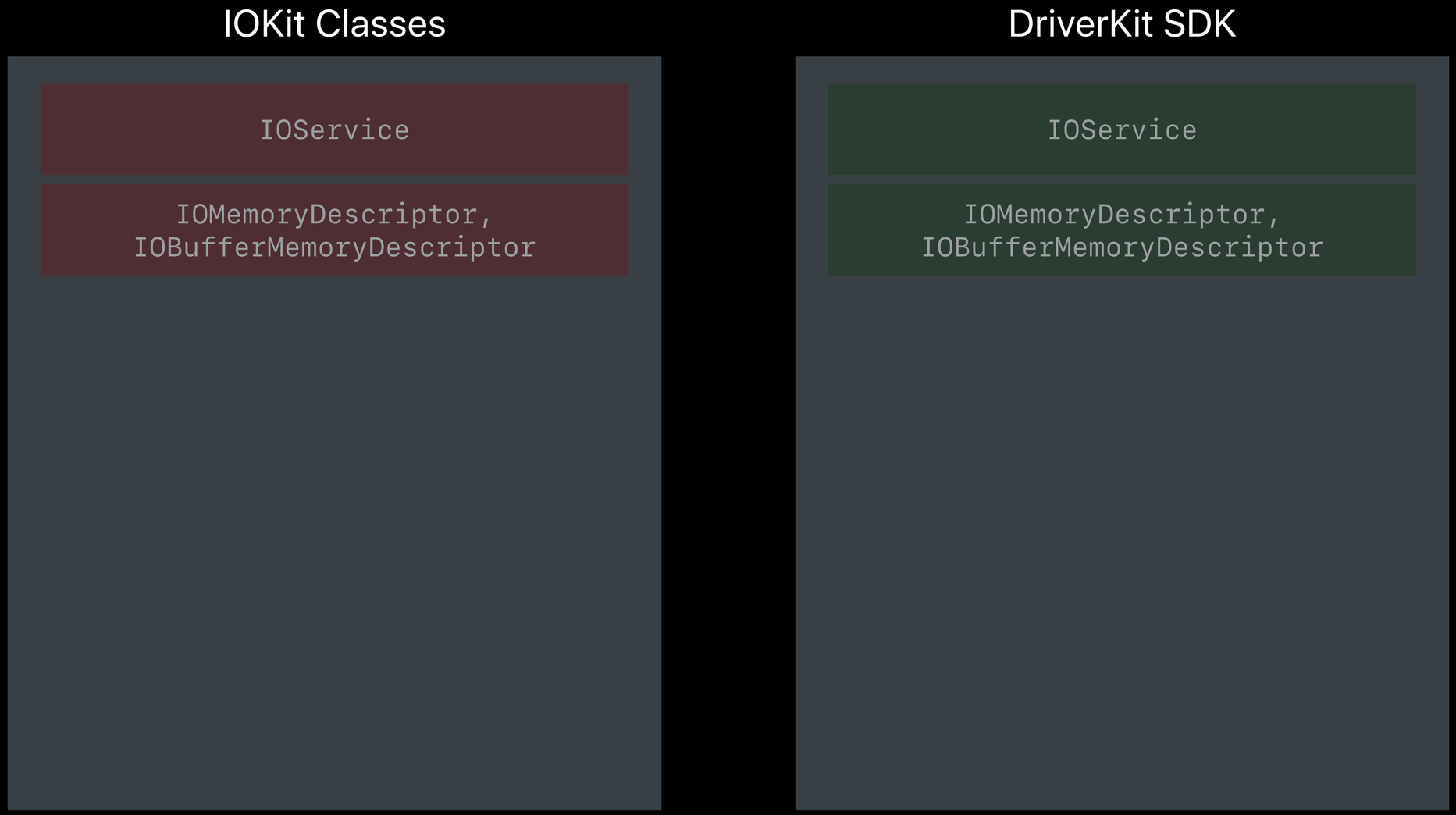
DriverKit SDK

Classes



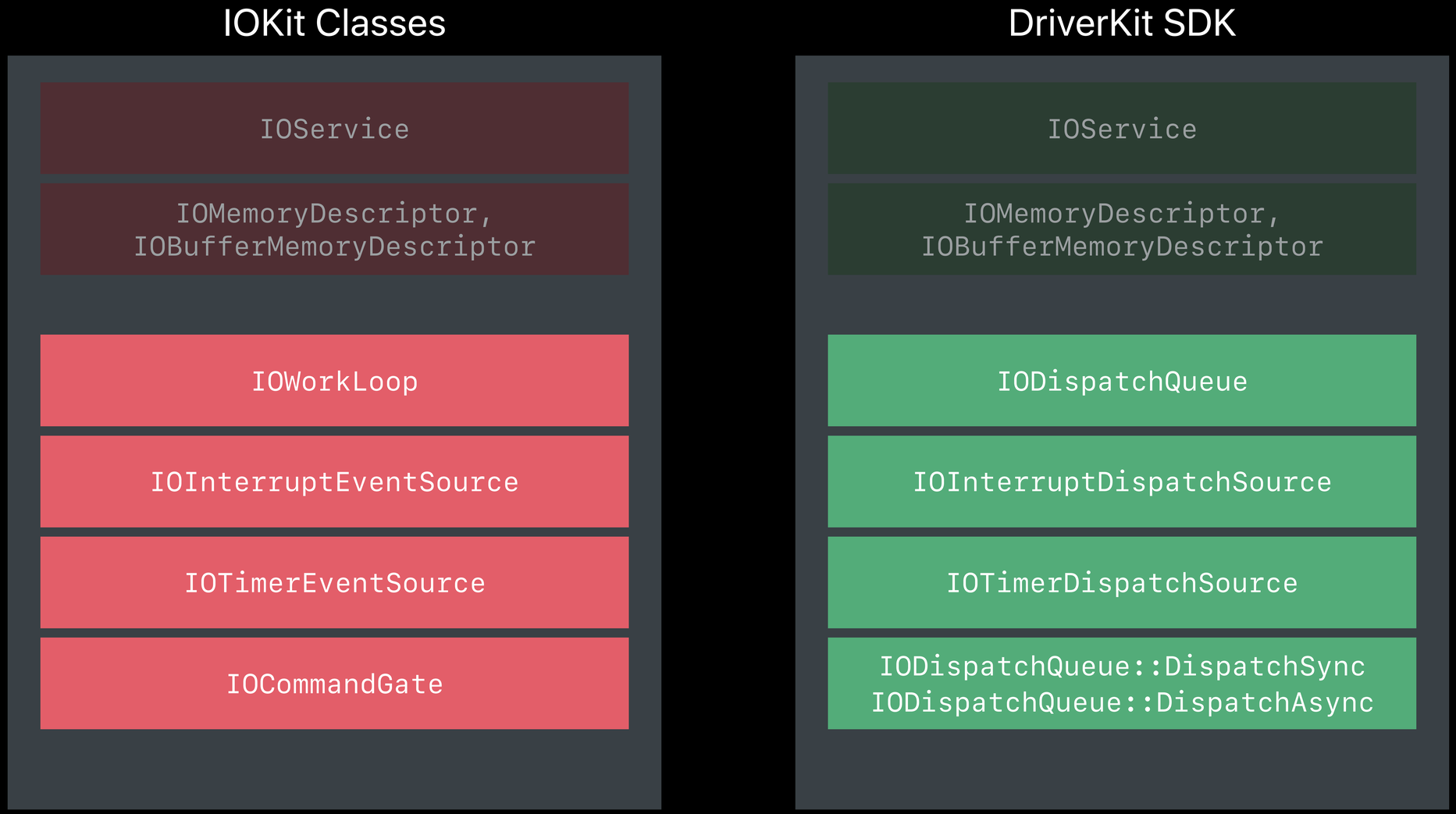
DriverKit SDK

Classes



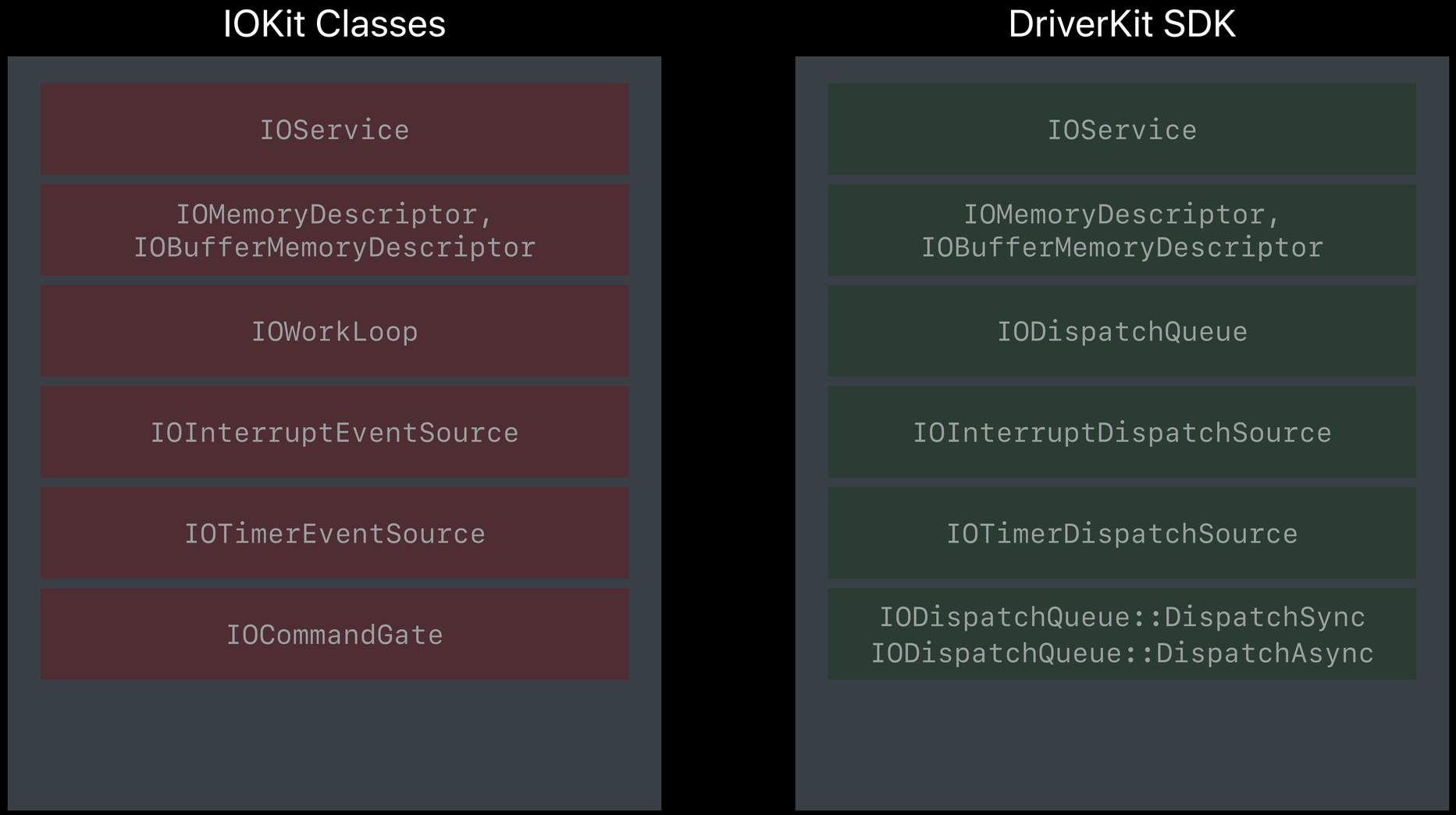
DriverKit SDK

Classes



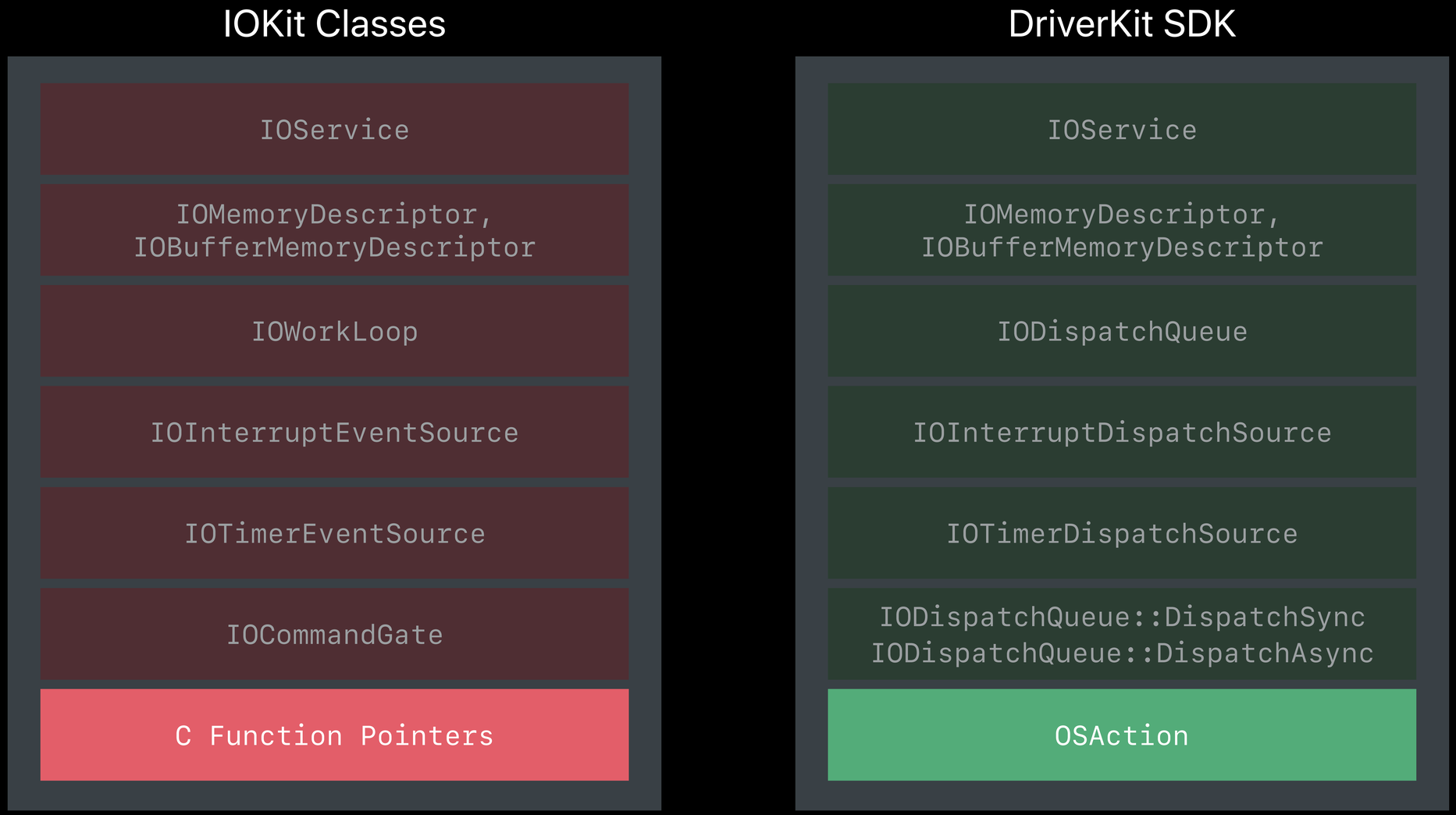
DriverKit SDK

Classes



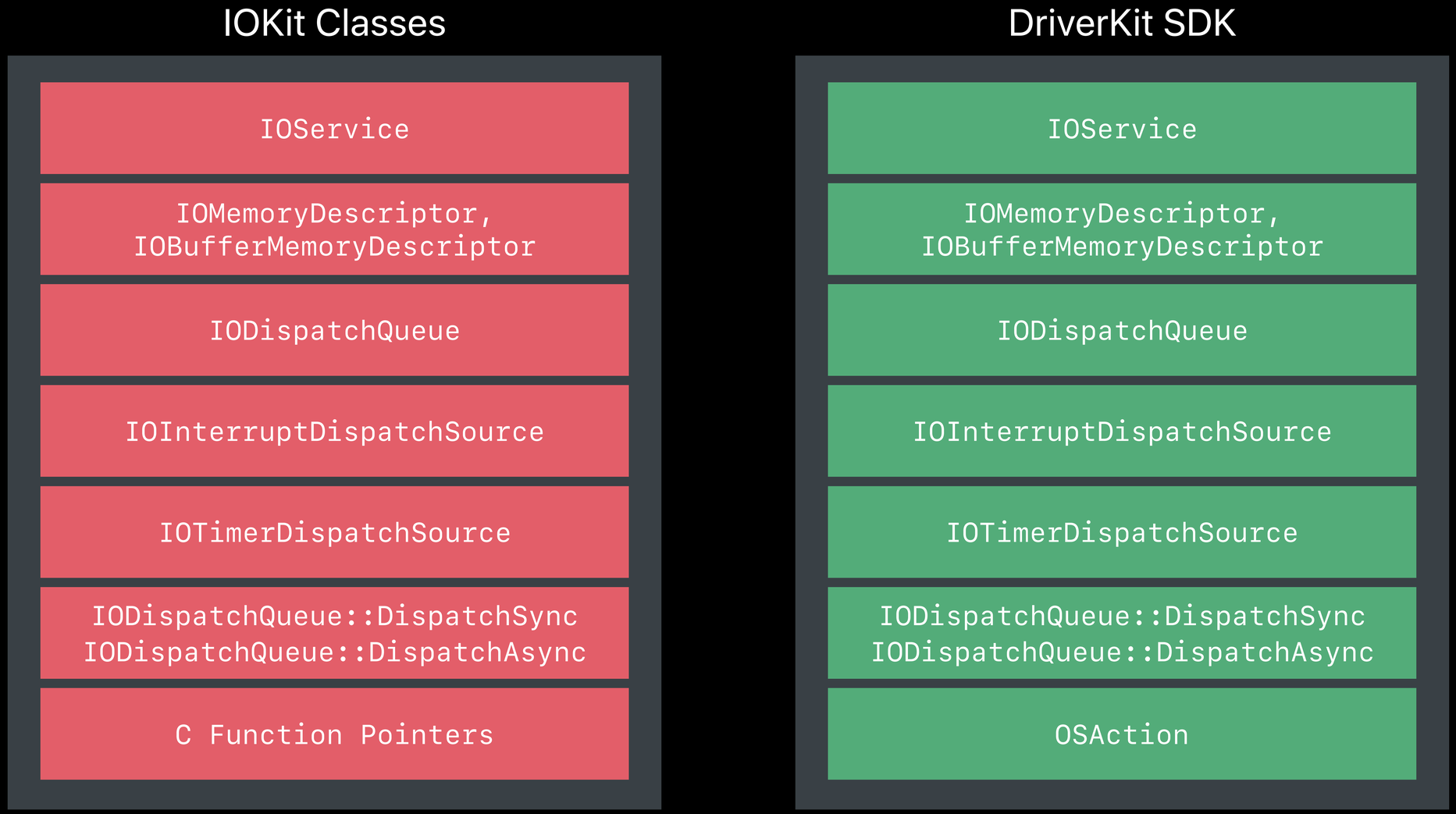
DriverKit SDK

Classes



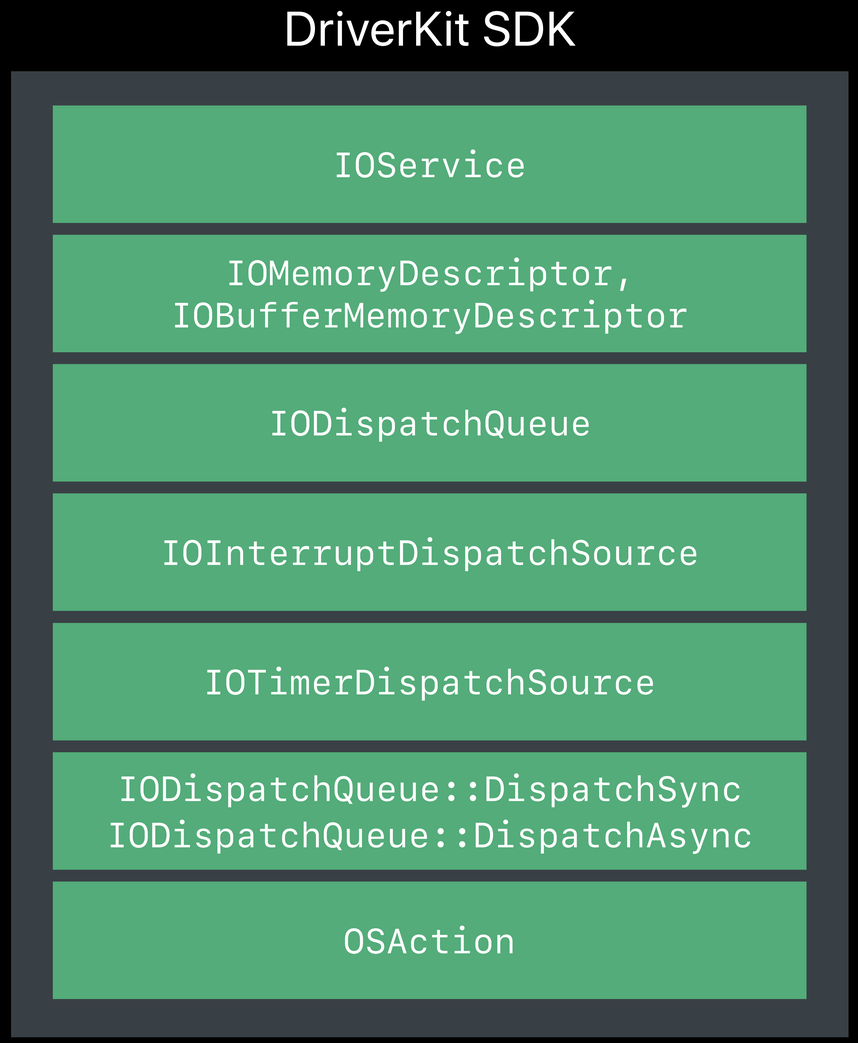
DriverKit SDK

Classes



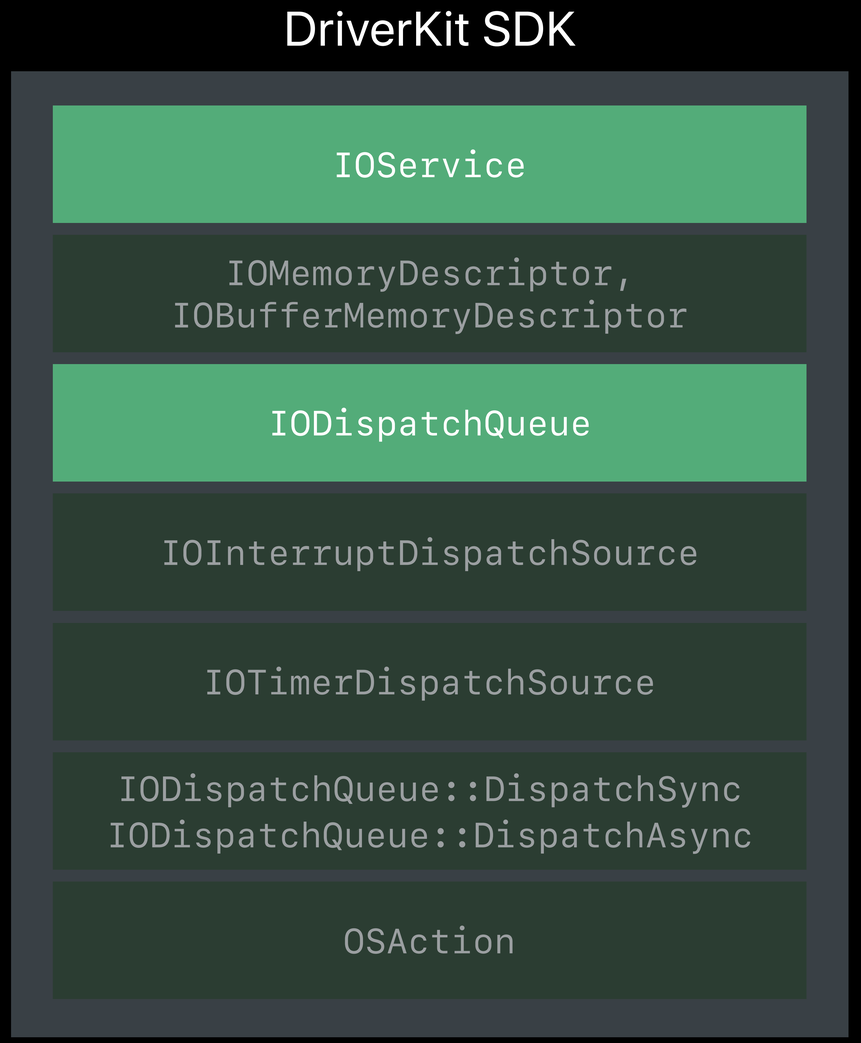
DriverKit SDK

Classes



DriverKit SDK

IOService

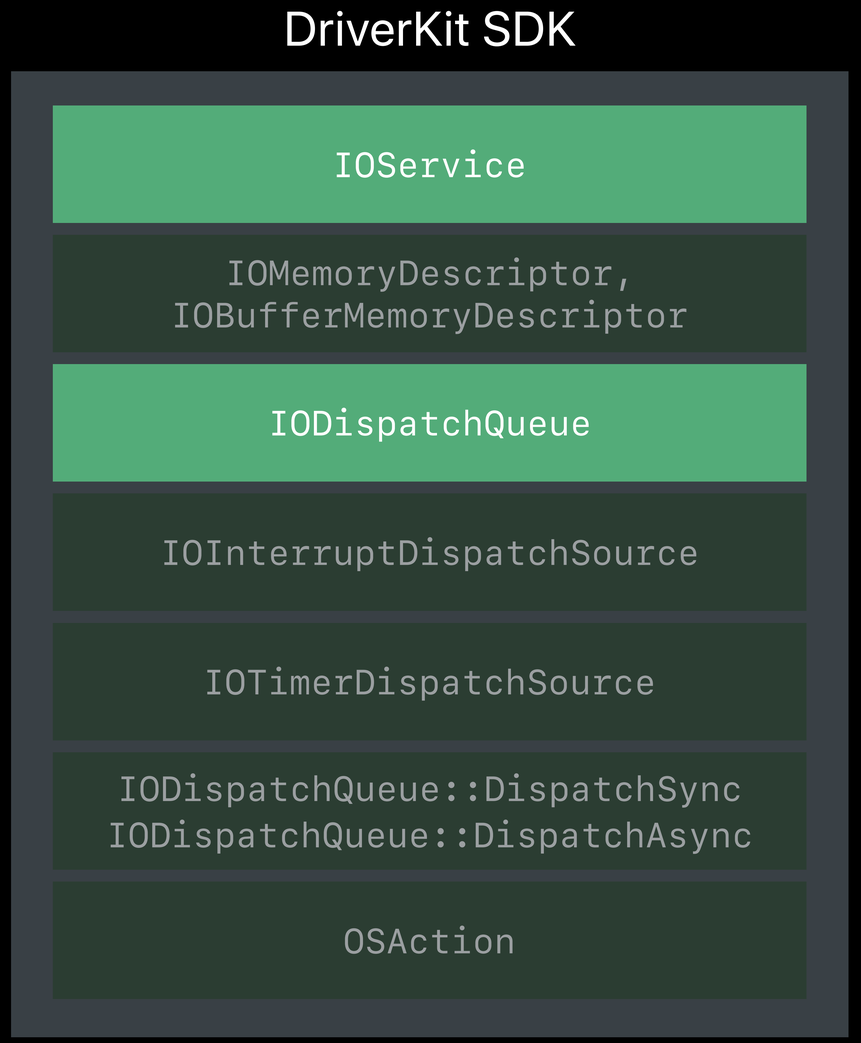


DriverKit SDK

IOService

IOService lifecycle APIs from IOKit

- Start/Stop/Terminate



DriverKit SDK

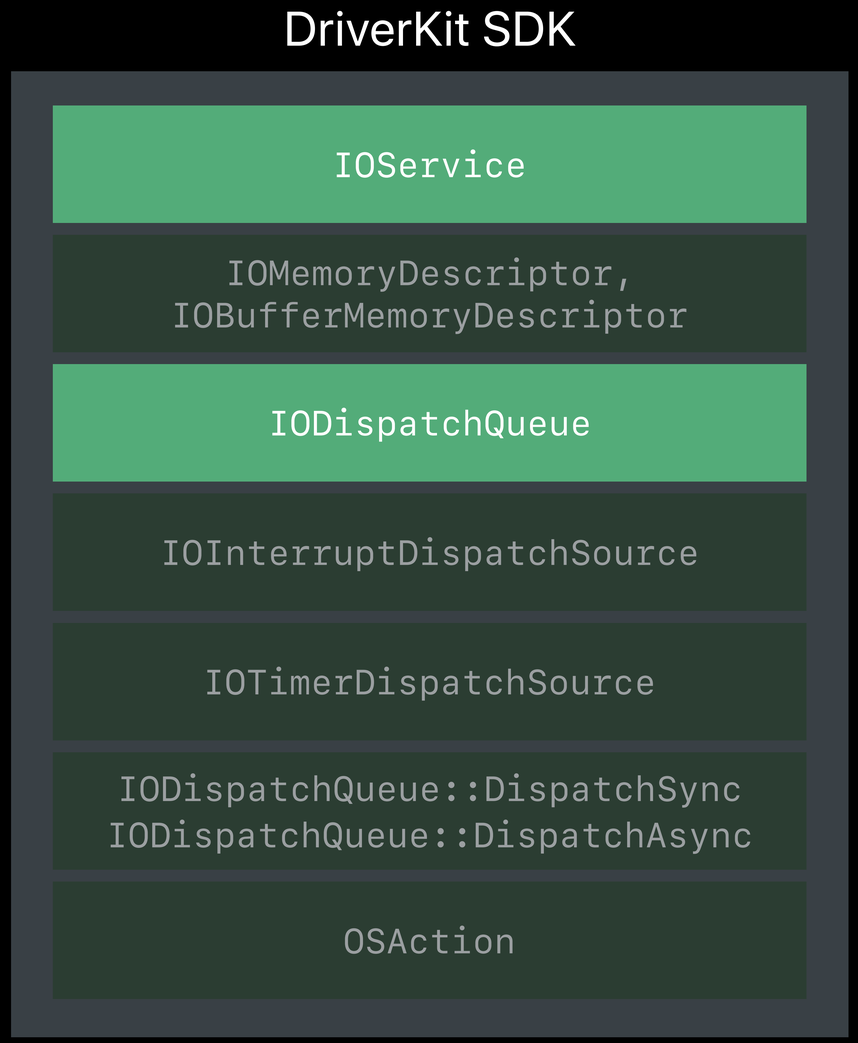
IOService

IOService lifecycle APIs from IOKit

- Start/Stop/Terminate

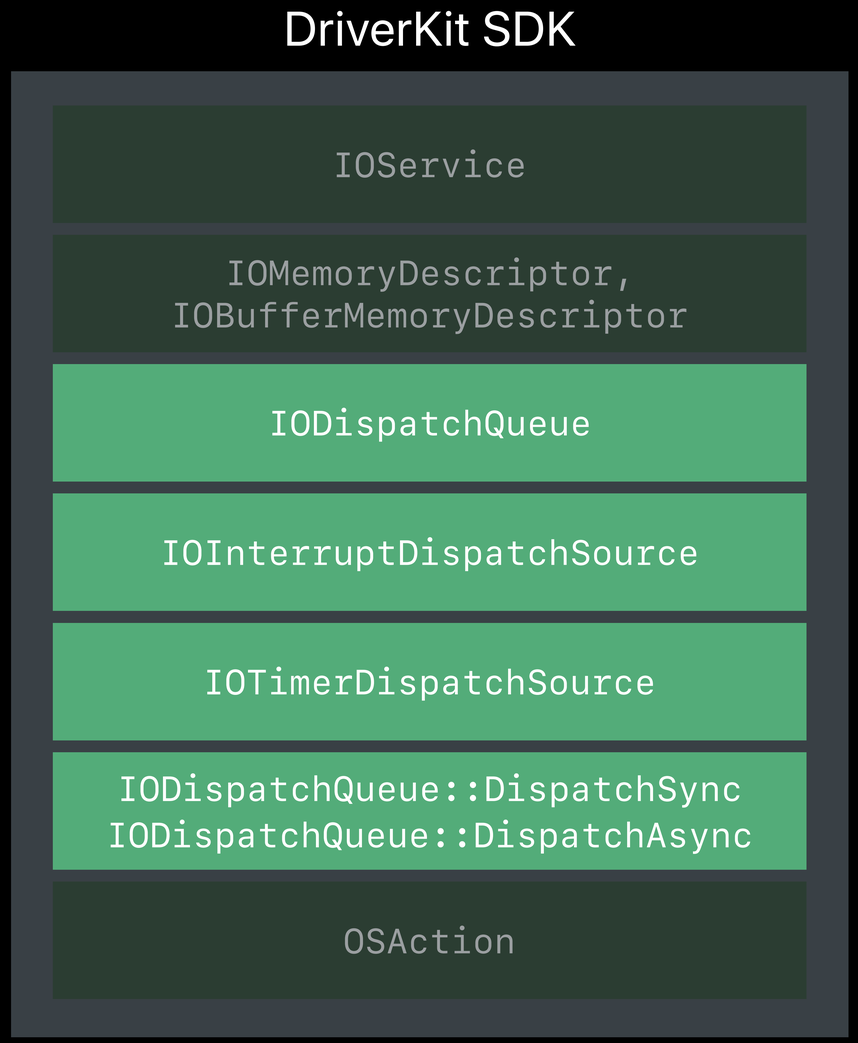
IODispatchQueue from Grand Central Dispatch

- All methods invoked on a queue
- Drivers have control of their queues



DriverKit SDK

Events



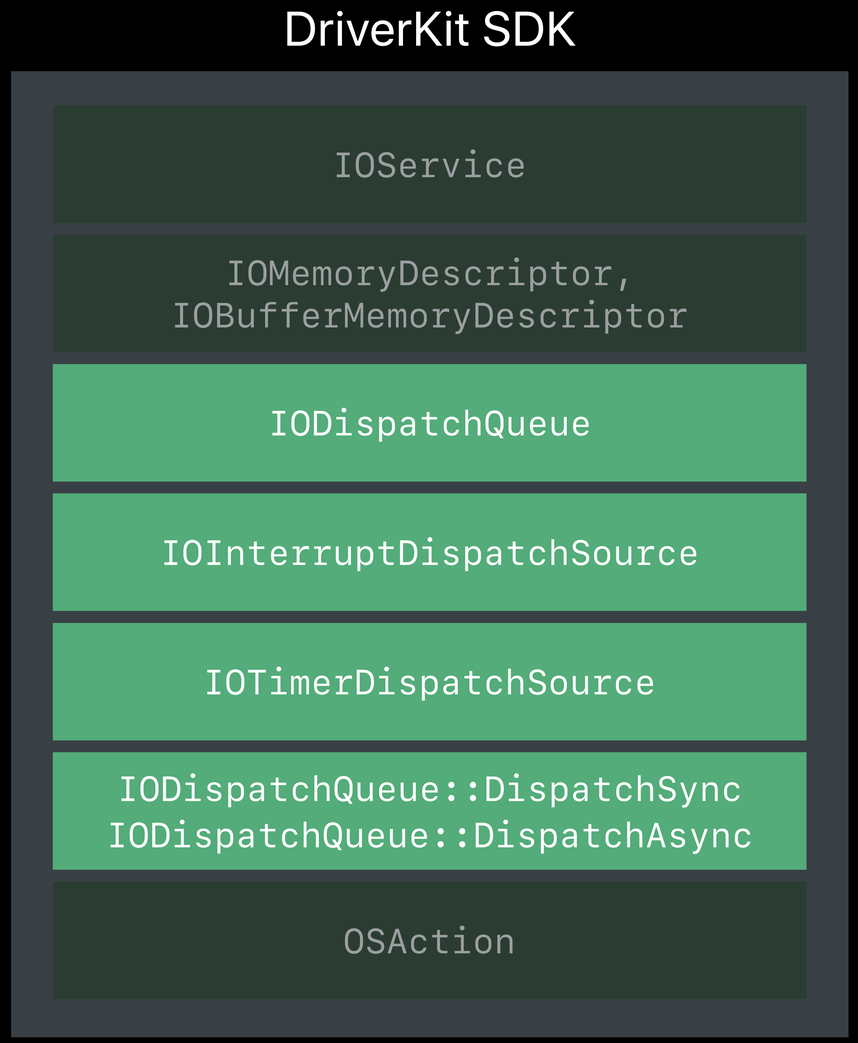
DriverKit SDK

Events

Equivalent to IOWorkLoop model in IOKit

Event handling with GCD

- Queues and dispatch sources



DriverKit SDK

Events

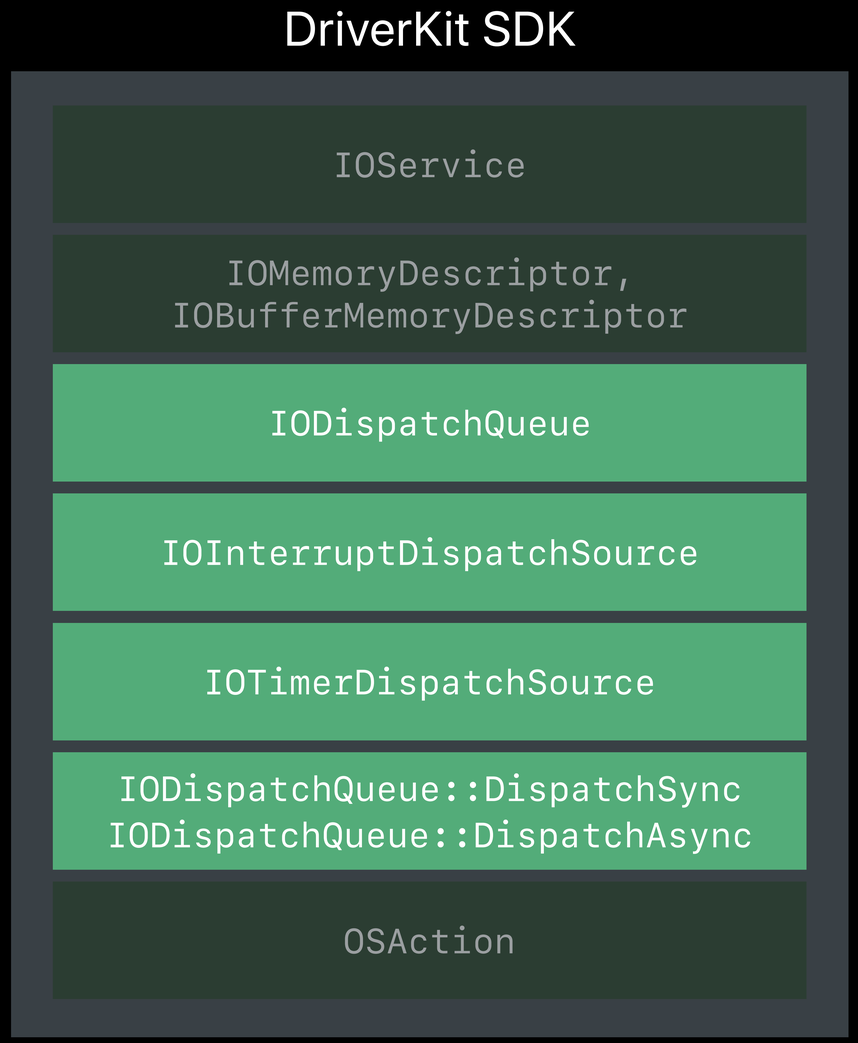
Equivalent to IOWorkLoop model in IOKit

Event handling with GCD

- Queues and dispatch sources

+ IOSharedDataQueueDispatchSource

- Shared memory ring buffer



DriverKit SDK

OSAction

DriverKit SDK

OSAction

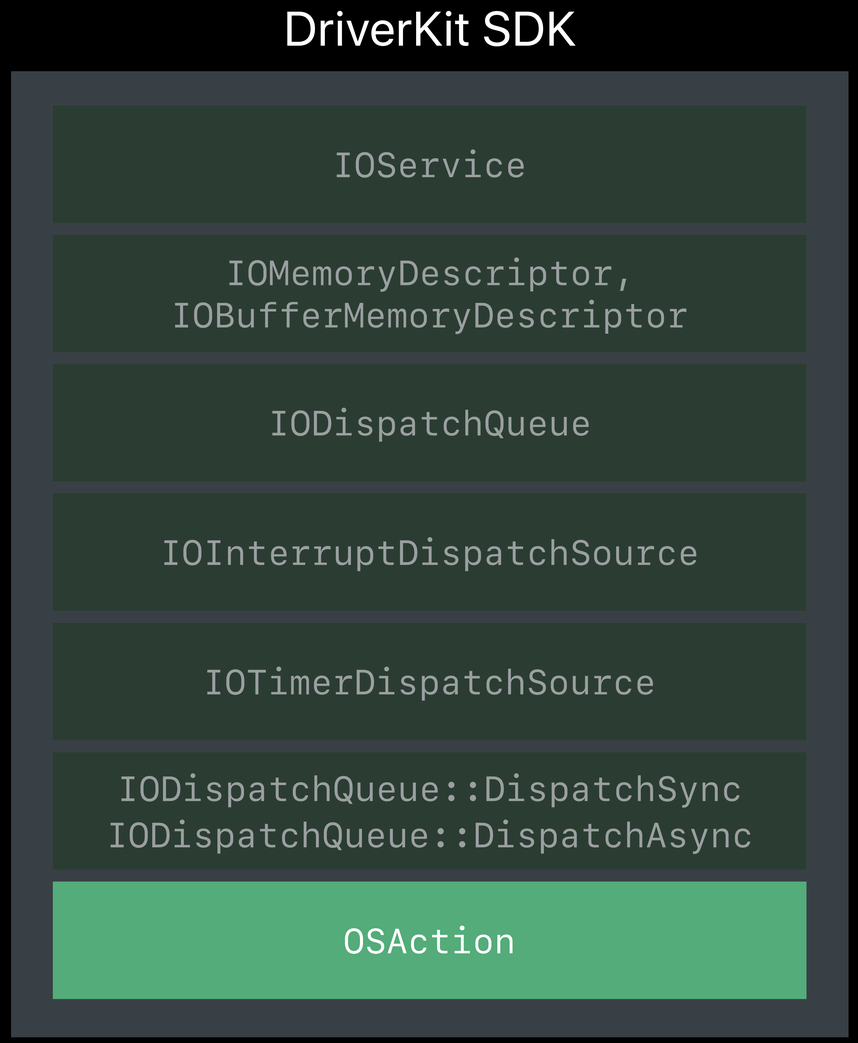
Async callbacks use OSAction

DriverKit SDK

OSAction

Async callbacks use OSAction

- Holds private state for the caller

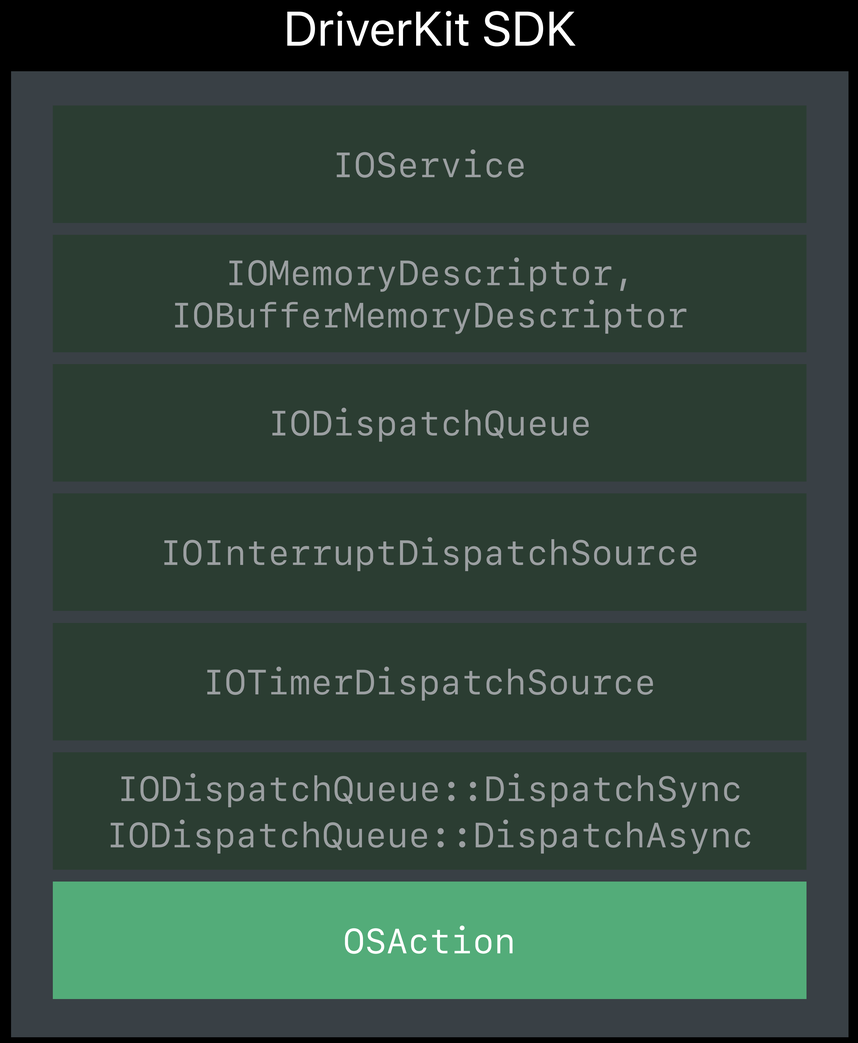


DriverKit SDK

OSAction

Async callbacks use OSAction

- Holds private state for the caller
- Arbitrary arguments with type checking



DriverKit SDK

Interface Definitions

NEW

DriverKit SDK

Interface Definitions

NEW

New filetype: **.iig** defines an interface as a C++ class

DriverKit SDK

Interface Definitions

NEW

New filetype: **.iig** defines an interface as a C++ class

Processed by **IOKit Interface Generator** tool — iig

DriverKit SDK

Interface Definitions

NEW

New filetype: **.iig** defines an interface as a C++ class

Processed by **IOKit Interface Generator** tool — iig

Standard C/C++ types and structure definitions

DriverKit SDK

Interface Definitions

NEW

New filetype: **.iig** defines an interface as a C++ class

Processed by **IOKit Interface Generator** tool — iig

Standard C/C++ types and structure definitions

New attributes for messaging, dispatch queues

```
// DriverKit SDK: Interface Definitions



class KERNEL IOService : public OSObject
{
public:
    virtual kern_return_t Start(IOService *provider) LOCAL;
    virtual kern_return_t Stop(IOService *provider)  LOCAL;
};
```

```
// DriverKit SDK: Interface Definitions
```

KERNEL : class is implemented in the kernel

```
class KERNEL IOService : public OSObject
{
public:
    virtual kern_return_t Start(IOService *provider) LOCAL;
    virtual kern_return_t Stop(IOService *provider)  LOCAL;
};
```

```
// DriverKit SDK: Interface Definitions
```

KERNEL : class is implemented in the kernel

```
class KERNEL IOService : public OSObject
{
public:
    virtual kern_return_t Start(IOService *provider) LOCAL;
    virtual kern_return_t Stop(IOService *provider)  LOCAL;
};
```

LOCAL : method is implemented in the driver

DriverKit SDK

Families

DriverKit SDK

Families

NetworkingDriverKit

DriverKit SDK

Families

NetworkingDriverKit

HIDDriverKit

DriverKit SDK

Families

NetworkingDriverKit

HIDDriverKit

USBSerialDriverKit

DriverKit SDK

Families

NetworkingDriverKit

HIDDriverKit

USBSerialDriverKit

USBDriverKit

Driver Extension Security

Entitlements

Entitlements

Entitlements required

Entitlements

Entitlements required

- All Driver Extensions: `com.apple.developer.driverkit`

Entitlements

Entitlements required

- All Driver Extensions: `com.apple.developer.driverkit`
- To attach to a device (transport entitlement, specific to device type)
  e.g. `com.apple.developer.driverkit.transport.usb`

Entitlements

Entitlements required

- All Driver Extensions: `com.apple.developer.driverkit`
- To attach to a device (transport entitlement, specific to device type)
  e.g. `com.apple.developer.driverkit.transport.usb`
- To provide a service to the OS (family entitlement)
  e.g. `com.apple.developer.driverkit.family.hid.device`

Driver Extension Compatibility

Driver Extension Compatibility

Install a Kernel Extension only on Mojave or earlier

Use SystemExtensions framework to provide a Driver Extension on macOS 10.15

Example

Creating a USB driver with USBDriverKit.framework

Scott Deandrea, Core Transports

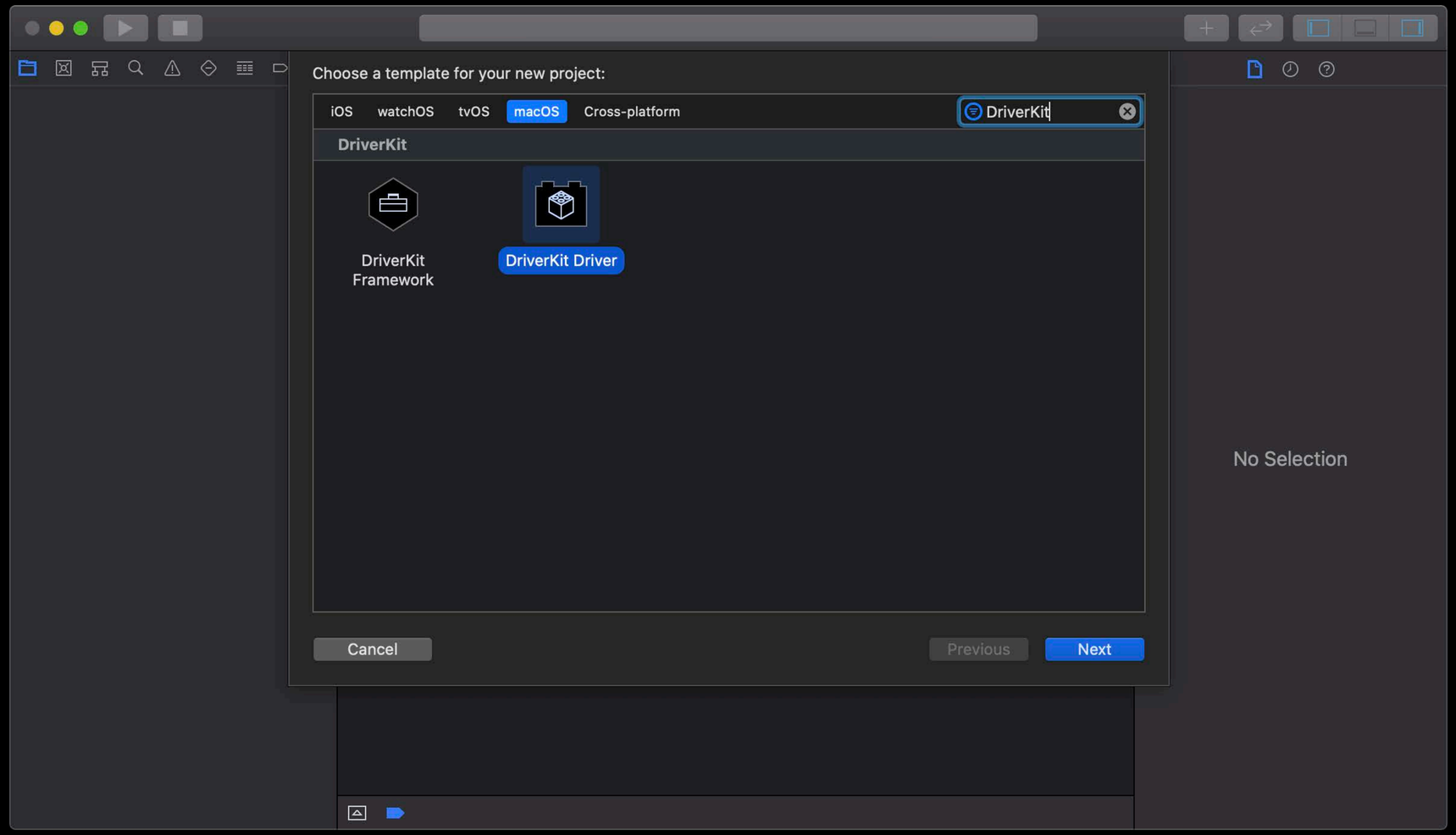
DriverKit Xcode template

MyUserUSBInterfaceDriver class definition

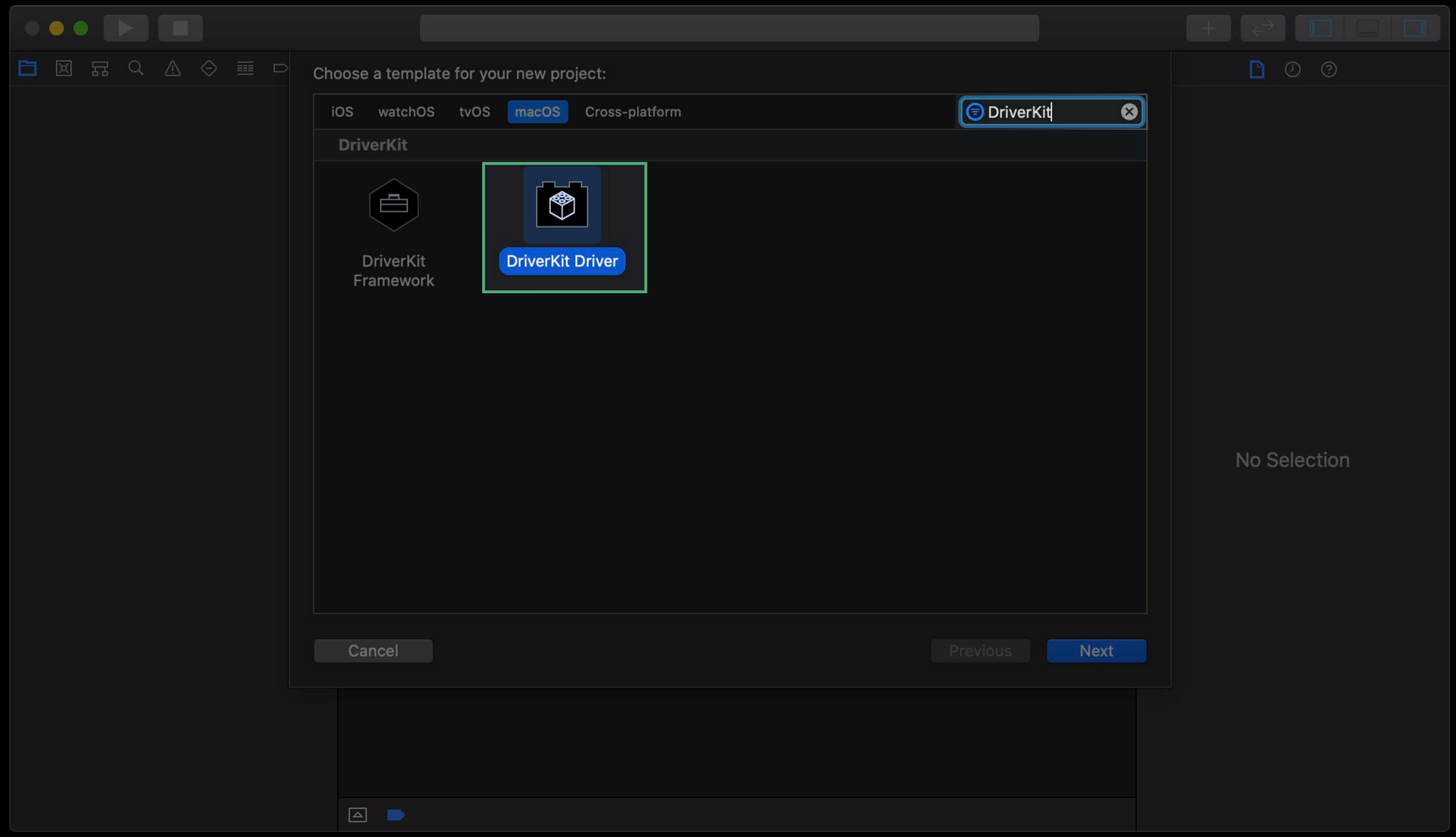
MyUserUSBInterfaceDriver implementation

Demo

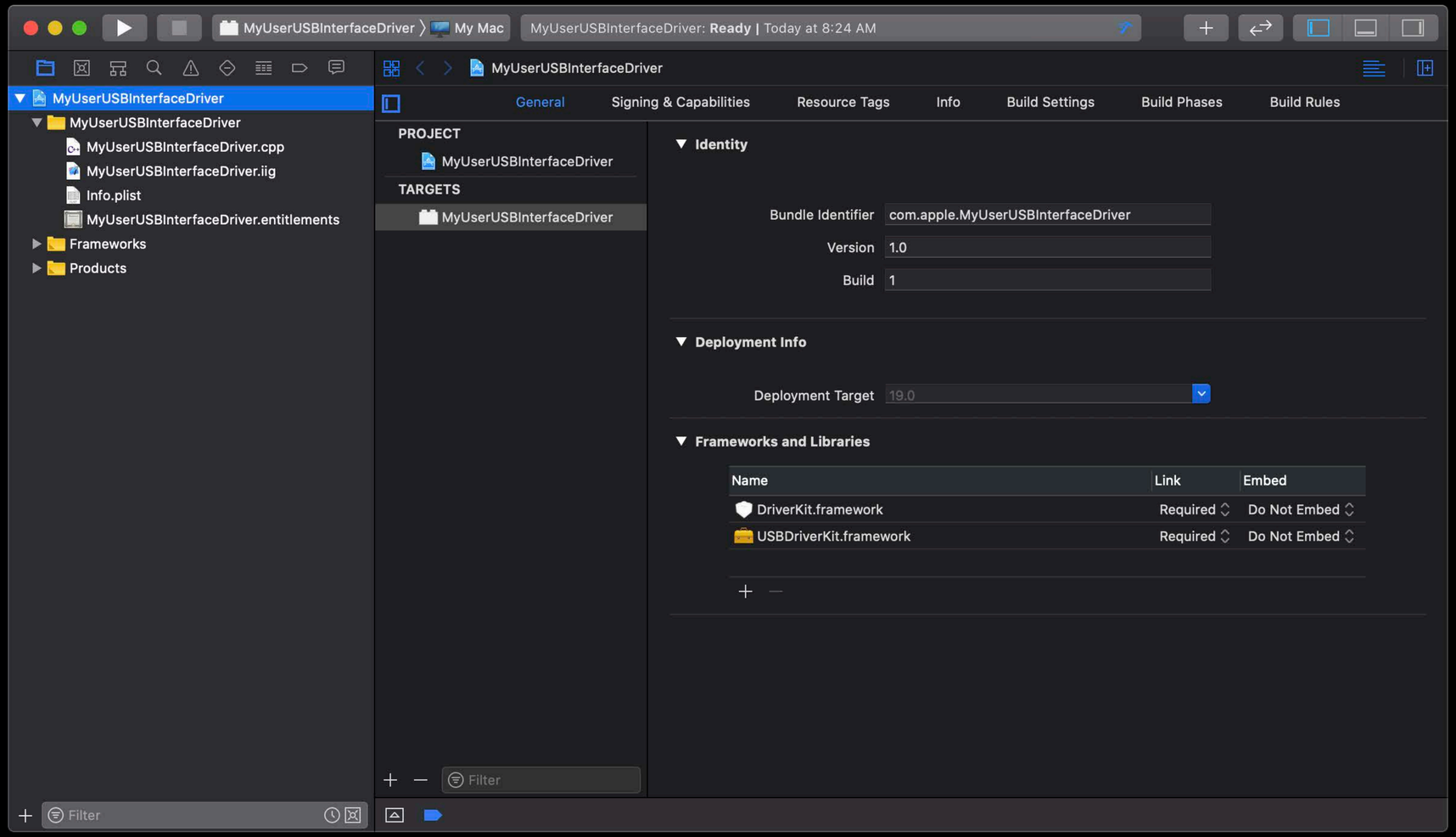
Using Xcode's DriverKit Template



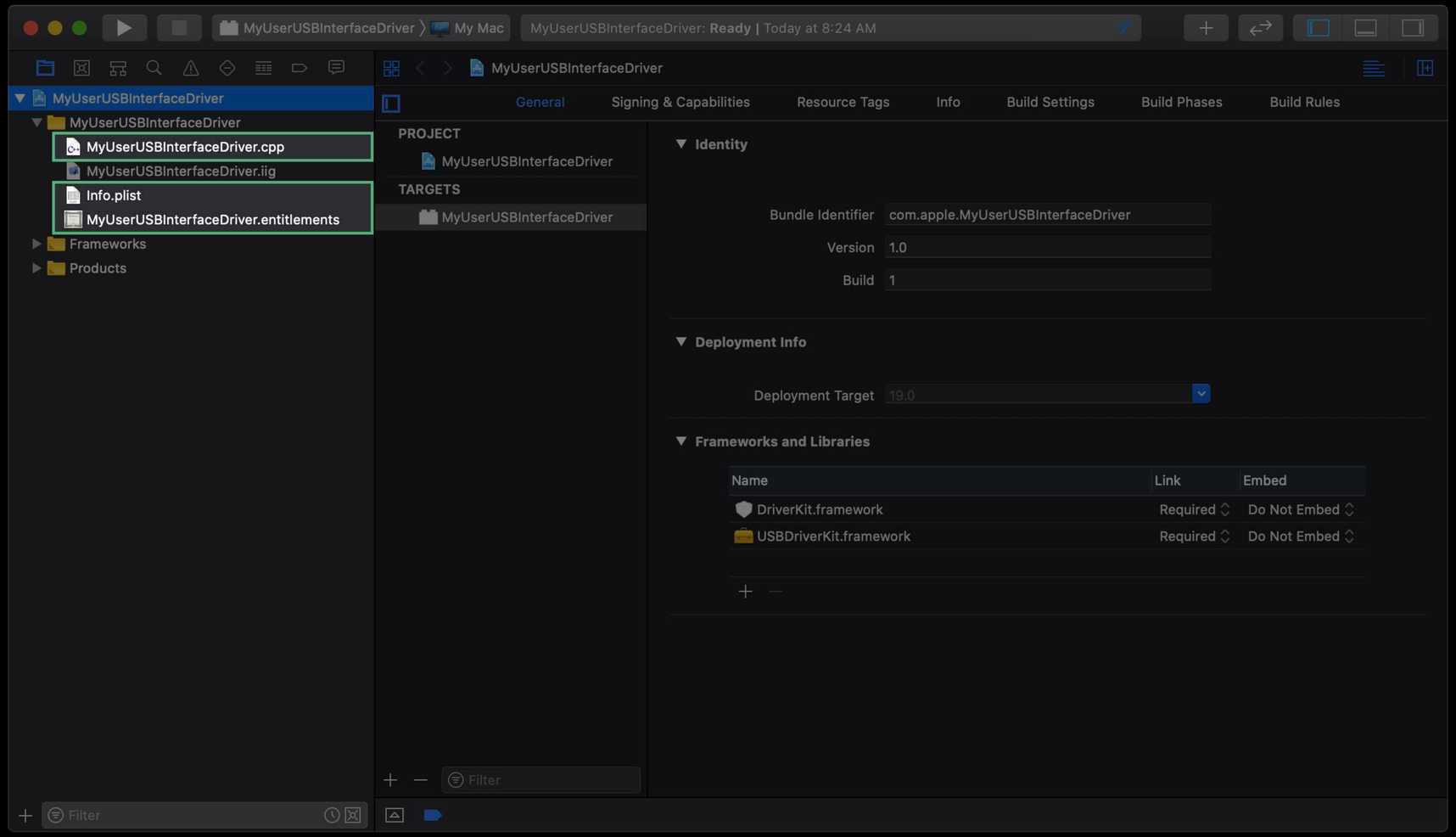
Using Xcode's DriverKit Template



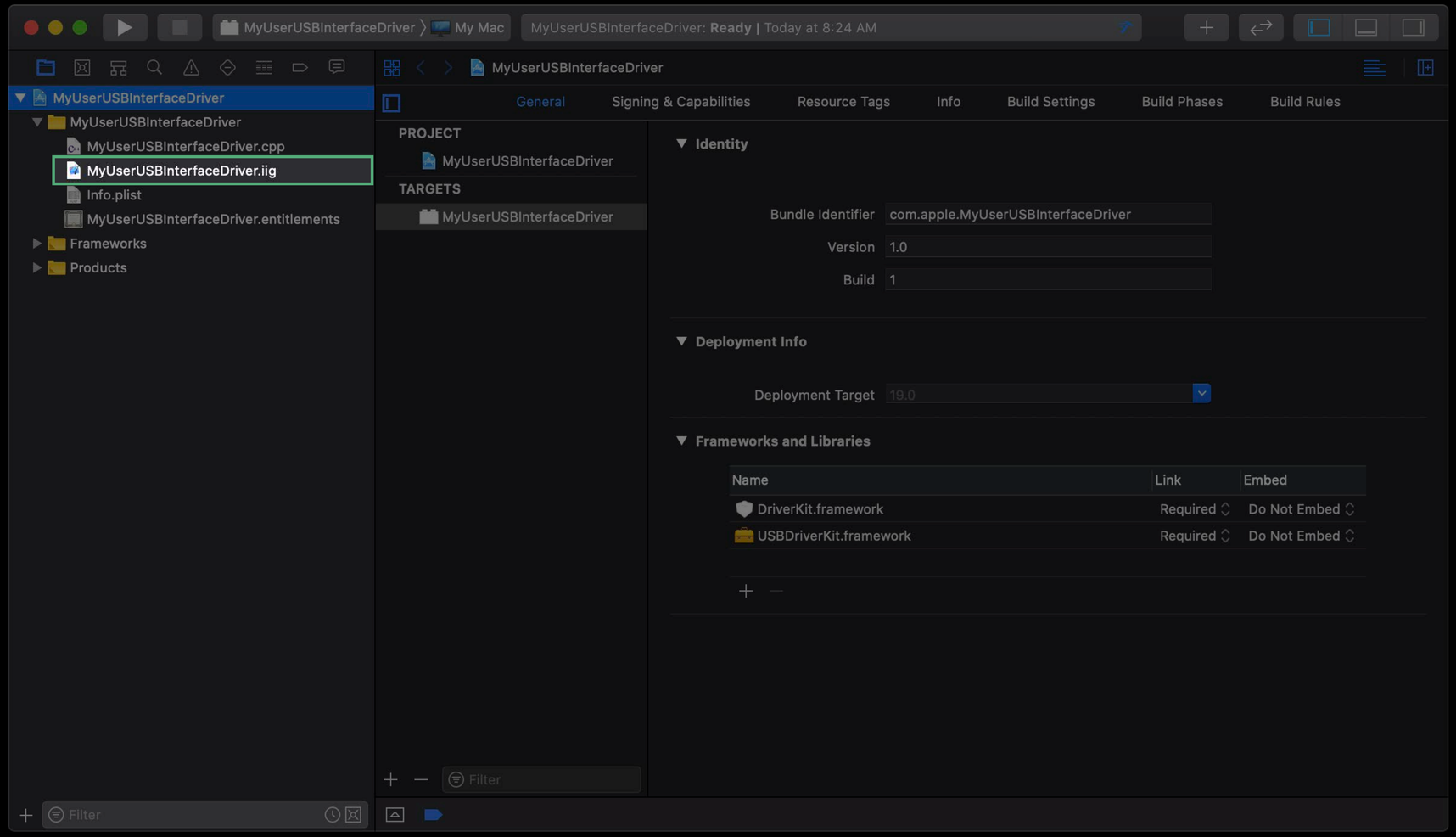
Using Xcode's DriverKit Template



Using Xcode's DriverKit Template



Using Xcode's DriverKit Template



```
// MyUserUSBInterfaceDriver.iig

public:
    virtual bool init () override;
    virtual kern_return_t Start (IOService *provider) override;
    virtual kern_return_t Stop (IOService *provider) override;
    virtual void free () override;

protected:
    virtual void ReadComplete (OSAction *action,
                               IOReturn  status,
                               uint32_t  actualByteCount,
                               uint64_t  completionTimestamp)
                               TYPE(IOUSBHostPipe::CompleteAsyncIO);
```

```
// MyUserUSBInterfaceDriver.iig

public:
    virtual bool init () override;
    virtual kern_return_t Start (IOService *provider) override;
    virtual kern_return_t Stop (IOService *provider) override;
    virtual void free () override;

protected:
    virtual void ReadComplete (OSAction *action,
                               IOReturn  status,
                               uint32_t  actualByteCount,
                               uint64_t  completionTimestamp)
                               TYPE(IOUSBHostPipe::CompleteAsyncIO);
```

```
// MyUserUSBInterfaceDriver.iig

public:
    virtual bool init () override;
    virtual kern_return_t Start (IOService *provider) override;
    virtual kern_return_t Stop (IOService *provider) override;
    virtual void free () override;

protected:
    virtual void ReadComplete (OSAction *action,
                               IOReturn  status,
                               uint32_t  actualByteCount,
                               uint64_t  completionTimestamp)
                               TYPE(IOUSBHostPipe::CompleteAsyncIO);
```

```
// MyUserUSBInterfaceDriver.cpp Instance Variable Definition

struct MyUserUSBInterfaceDriver_IVars
{
    IOUSBHostInterface        *interface;
    IOUSBHostPipe             *inPipe;
    OSAction                  *ioCompleteCallback;
    IOBufferMemoryDescriptor *inData;
    uint16_t                   maxPacketSize;
};
```

```
// MyUserUSBInterfaceDriver.cpp Instance Variable Definition

struct MyUserUSBInterfaceDriver_IVars
{
    IOUSBHostInterface        *interface;
    IOUSBHostPipe             *inPipe;
    OSAction                  *ioCompleteCallback;
    IOBufferMemoryDescriptor *inData;
    uint16_t                   maxPacketSize;
};
```

```
// MyUserUSBInterfaceDriver.cpp Instance Variable Definition

struct MyUserUSBInterfaceDriver_IVars
{
    IOUSBHostInterface        *interface;
    IOUSBHostPipe             *inPipe;
    OSAction                  *ioCompleteCallback;
    IOBufferMemoryDescriptor *inData;
    uint16_t                   maxPacketSize;
};
```

```
// MyUserUSBInterfaceDriver.cpp Instance Variable Definition

struct MyUserUSBInterfaceDriver_IVars
{
    IOUSBHostInterface        *interface;
    IOUSBHostPipe             *inPipe;
    OSAction                  *ioCompleteCallback;
    IOBufferMemoryDescriptor *inData;
    uint16_t                   maxPacketSize;
};
```

```
// MyUserUSBInterfaceDriver.cpp init () method

bool
MyUserUSBInterfaceDriver::init ()
{
    bool result = false;

    result = super::init();
    __Require(true == result, Exit);

    ivars = IONewZero(MyUserUSBInterfaceDriver_IVars, 1);
    __Require_Action(NULL != ivars, Exit, result = false);

Exit:
    return result;
}
```

```
// MyUserUSBInterfaceDriver.cpp init () method

bool
MyUserUSBInterfaceDriver::init ()
{
    bool result = false;

    result = super::init();
    __Require(true == result, Exit);

    ivars = IONewZero(MyUserUSBInterfaceDriver_IVars, 1);
    __Require_Action(NULL != ivars, Exit, result = false);

Exit:
    return result;
}
```

```
// MyUserUSBInterfaceDriver.cpp init () method

bool
MyUserUSBInterfaceDriver::init ()
{
    bool result = false;

    result = super::init();
    __Require(true == result, Exit);

    ivars = IONewZero(MyUserUSBInterfaceDriver_IVars, 1);
    __Require_Action(NULL != ivars, Exit, result = false);

Exit:
    return result;
}
```

```
// MyUserUSBInterfaceDriver.cpp Start () method

kern_return_t
IMPL (MyUserUSBInterfaceDriver,
      Start)
{
    kern_return_t                   ret;
    IOUSBStandardEndpointDescriptors descriptors;

    ret = Start(provider, SUPERDISPATCH);
    __Require(kIOReturnSuccess == ret, Exit);

    ivars->interface = OSDynamicCast(IOUSBHostInterface, provider);
    __Require_Action(NULL != ivars->interface, Exit, ret = kIOReturnNoDevice);
…
```

```
// MyUserUSBInterfaceDriver.cpp Start () method

kern_return_t
IMPL (MyUserUSBInterfaceDriver,
      Start)
{
    kern_return_t                   ret;
    IOUSBStandardEndpointDescriptors descriptors;

    ret = Start(provider, SUPERDISPATCH);
    __Require(kIOReturnSuccess == ret, Exit);

    ivars->interface = OSDynamicCast(IOUSBHostInterface, provider);
    __Require_Action(NULL != ivars->interface, Exit, ret = kIOReturnNoDevice);
…
```

```
// MyUserUSBInterfaceDriver.cpp Start () method

kern_return_t
IMPL (MyUserUSBInterfaceDriver,
      Start)
{
    kern_return_t                   ret;
    IOUSBStandardEndpointDescriptors descriptors;

    ret = Start(provider, SUPERDISPATCH);
    __Require(kIOReturnSuccess == ret, Exit);

    ivars->interface = OSDynamicCast(IOUSBHostInterface, provider);
    __Require_Action(NULL != ivars->interface, Exit, ret = kIOReturnNoDevice);
…
```

```
// MyUserUSBInterfaceDriver.cpp Start () method

kern_return_t
IMPL (MyUserUSBInterfaceDriver,
      Start)
{
    kern_return_t                   ret;
    IOUSBStandardEndpointDescriptors descriptors;

    ret = Start(provider, SUPERDISPATCH);
    __Require(kIOReturnSuccess == ret, Exit);

    ivars->interface = OSDynamicCast(IOUSBHostInterface, provider);
    __Require_Action(NULL != ivars->interface, Exit, ret = kIOReturnNoDevice);
…
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = ivars->interface->Open(this, 0, NULL);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CopyPipe(kMyEndpointAddress, &ivars->inPipe);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CreateIOBuffer(kIOMemoryDirectionIn,
                                       ivars->maxPacketSize,
                                       &ivars->inData);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = ivars->interface->Open(this, 0, NULL);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CopyPipe(kMyEndpointAddress, &ivars->inPipe);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CreateIOBuffer(kIOMemoryDirectionIn,
                                       ivars->maxPacketSize,
                                       &ivars->inData);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = ivars->interface->Open(this, 0, NULL);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CopyPipe(kMyEndpointAddress, &ivars->inPipe);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CreateIOBuffer(kIOMemoryDirectionIn,
                                       ivars->maxPacketSize,
                                       &ivars->inData);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = ivars->interface->Open(this, 0, NULL);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CopyPipe(kMyEndpointAddress, &ivars->inPipe);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CreateIOBuffer(kIOMemoryDirectionIn,
                                       ivars->maxPacketSize,
                                       &ivars->inData);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = ivars->interface->Open(this, 0, NULL);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CopyPipe(kMyEndpointAddress, &ivars->inPipe);
__Require(kIOReturnSuccess == ret, Exit);

ret = ivars->interface->CreateIOBuffer(kIOMemoryDirectionIn,
                                       ivars->maxPacketSize,
                                       &ivars->inData);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = OSAction::Create(this,
                       MyUserUSBInterfaceDriver_ReadComplete_ID,
                       IOUSBHostPipe_CompleteAsyncIO_ID,
                       0,
                       &ivars->ioCompleteCallback);
 __Require(kIOReturnSuccess == ret, Exit);

ret = ivars->inPipe->AsyncIO(ivars->inData,
                             ivars->maxPacketSize,
                             ivars->ioCompleteCallback,
                             0);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = OSAction::Create(this,
                       MyUserUSBInterfaceDriver_ReadComplete_ID,
                       IOUSBHostPipe_CompleteAsyncIO_ID,
                       0,
                       &ivars->ioCompleteCallback);
 __Require(kIOReturnSuccess == ret, Exit);

ret = ivars->inPipe->AsyncIO(ivars->inData,
                             ivars->maxPacketSize,
                             ivars->ioCompleteCallback,
                             0);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = OSAction::Create(this,
                       MyUserUSBInterfaceDriver_ReadComplete_ID,
                       IOUSBHostPipe_CompleteAsyncIO_ID,
                       0,
                       &ivars->ioCompleteCallback);
 __Require(kIOReturnSuccess == ret, Exit);

ret = ivars->inPipe->AsyncIO(ivars->inData,
                             ivars->maxPacketSize,
                             ivars->ioCompleteCallback,
                             0);
__Require(kIOReturnSuccess == ret, Exit);
```

```
// MyUserUSBInterfaceDriver.cpp Start () continued…


ret = OSAction::Create(this,
                       MyUserUSBInterfaceDriver_ReadComplete_ID,
                       IOUSBHostPipe_CompleteAsyncIO_ID,
                       0,
                       &ivars->ioCompleteCallback);
 __Require(kIOReturnSuccess == ret, Exit);

ret = ivars->inPipe->AsyncIO(ivars->inData,
                             ivars->maxPacketSize,
                             ivars->ioCompleteCallback,
                             0);
__Require(kIOReturnSuccess == ret, Exit);
```

Demo

System Extensions in Your Apps

Joe Auricchio, Darwin Runtime

System Extensions in Your Apps

System Extensions in Your Apps

How your extension relates to your app

System Extensions in Your Apps

How your extension relates to your app

Building and packaging the extension bundle

System Extensions in Your Apps

How your extension relates to your app

Building and packaging the extension bundle

Code signing and entitlements

System Extensions in Your Apps

How your extension relates to your app

Building and packaging the extension bundle

Code signing and entitlements

Installing, updating, uninstalling

A System Extension Is Always Part of an App

A System Extension Is Always Part of an App

No such thing as a “standalone system extension”

A System Extension Is Always Part of an App

No such thing as a “standalone system extension”

Users think in terms of apps

- Apps are what users purchase, install, and run
- System Extensions should be an implementation detail

A System Extension Is Always Part of an App

No such thing as a “standalone system extension”

Users think in terms of apps

- Apps are what users purchase, install, and run
- System Extensions should be an implementation detail

App is the user interface to control extension

- Using SystemExtensions framework — more in a moment

Distribute directly with Developer ID

Distribute directly with Developer ID

… or on the Mac App Store

System Extensions Should Be Identifiable

System Extensions Should Be Identifiable

User should recognize extension as part of your app

- Give a descriptive name with `CFBundleDisplayName` key
- Give a custom icon that relates to your app’s icon

System Extensions Should Be Identifiable

User should recognize extension as part of your app

- Give a descriptive name with `CFBundleDisplayName` key
- Give a custom icon that relates to your app’s icon

Include usage description in Info.plist

- What the extension does and why a user would run it
- Similar to other usage description keys (Camera, Calendars)

System Extensions Should Be Identifiable

User should recognize extension as part of your app

- Give a descriptive name with `CFBundleDisplayName` key
- Give a custom icon that relates to your app's icon

Include usage description in Info.plist

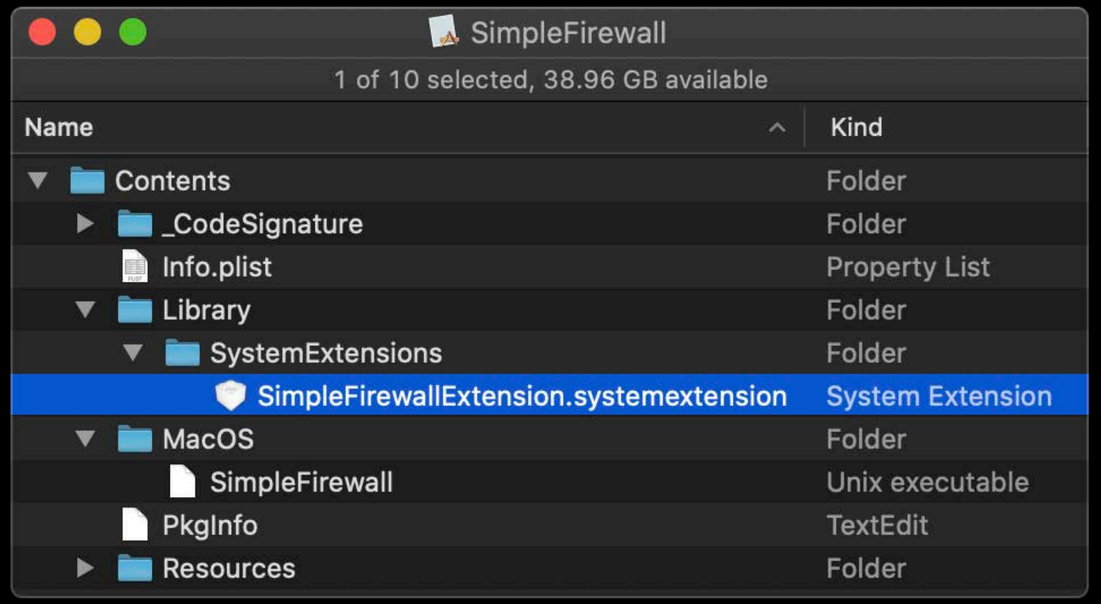
- What the extension does and why a user would run it
- Similar to other usage description keys (Camera, Calendars)

dexts: `OSBundleUsageDescription`, others: `NSSystemExtensionUsageDescription`

System Extension Bundles

System Extension Bundles

Embedded in application

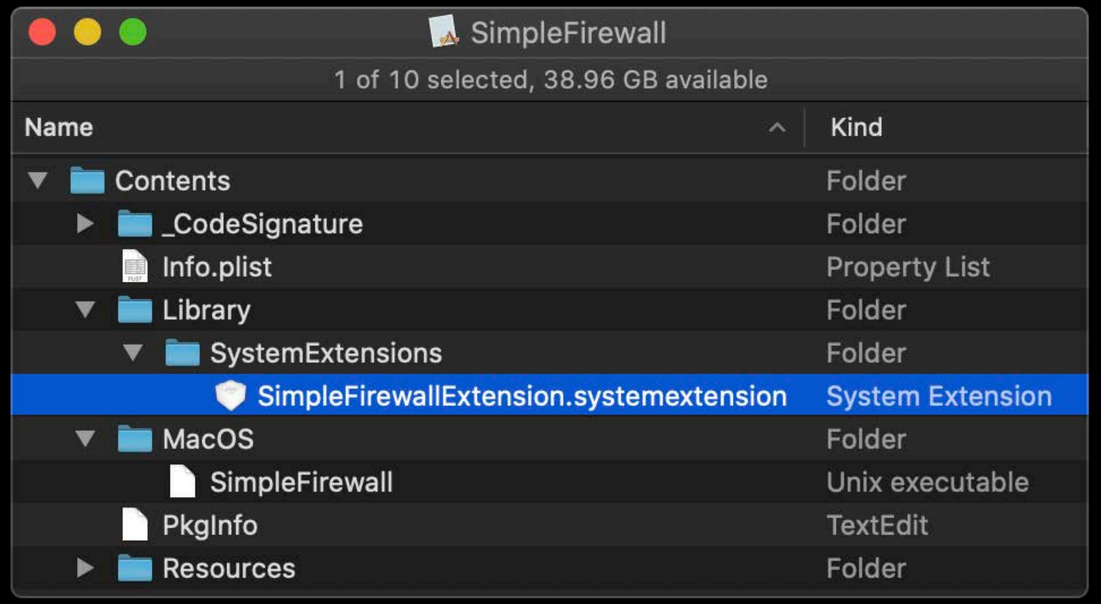


System Extension Bundles

Embedded in application

Driver Extensions

- `.dext` bundle resembling .kext
- `CFBundlePackageType = DEXT`
- Uses `OSBundle*` Info.plist keys
- Flat structure: no `Contents` folder



System Extension Bundles

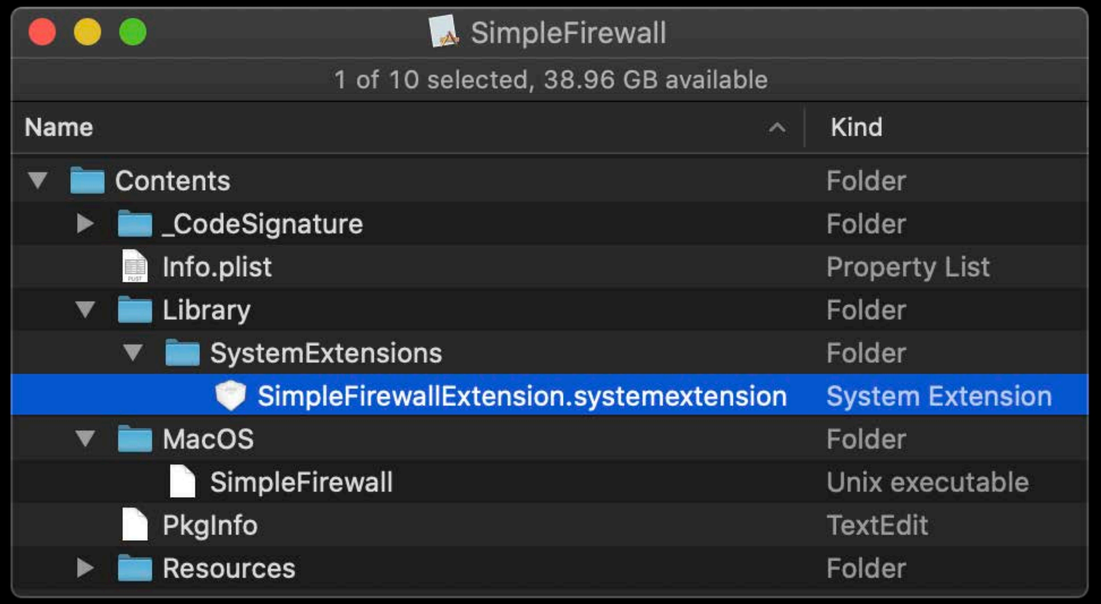
Embedded in application

Driver Extensions

- `.dext` bundle resembling .kext
- `CFBundlePackageType = DEXT`
- Uses `OSBundle*` Info.plist keys
- Flat structure: no `Contents` folder

Other extensions

- `.systemextension` bundle
- `CFBundlePackageType = SYSX`



Building in Xcode

Building in Xcode

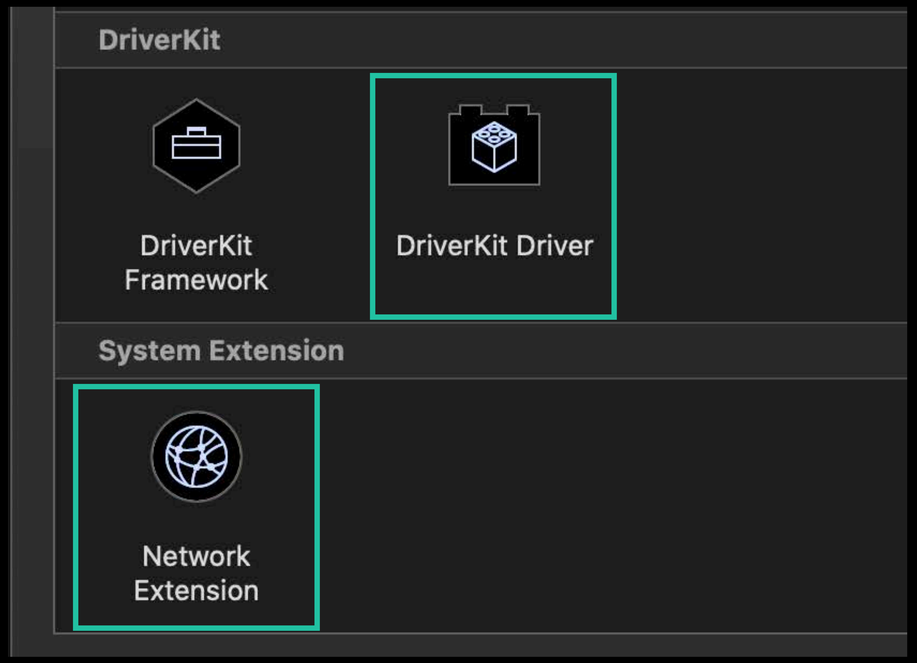
Build extension as another target in your project

Building in Xcode

Build extension as another target in your project

Xcode has templates for

- Network Extension
- DriverKit Driver



Building in Xcode

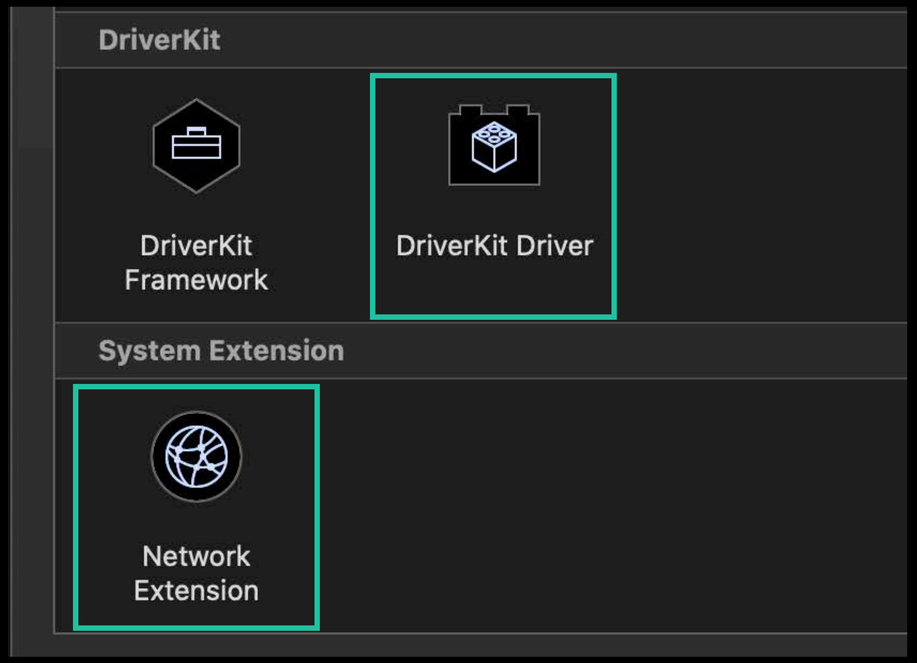
Build extension as another target in your project

Xcode has templates for

- Network Extension
- DriverKit Driver

Copy the extension into app bundle

- Copy Files Phase in app target
- `Contents/Library/SystemExtensions`



Building in Xcode

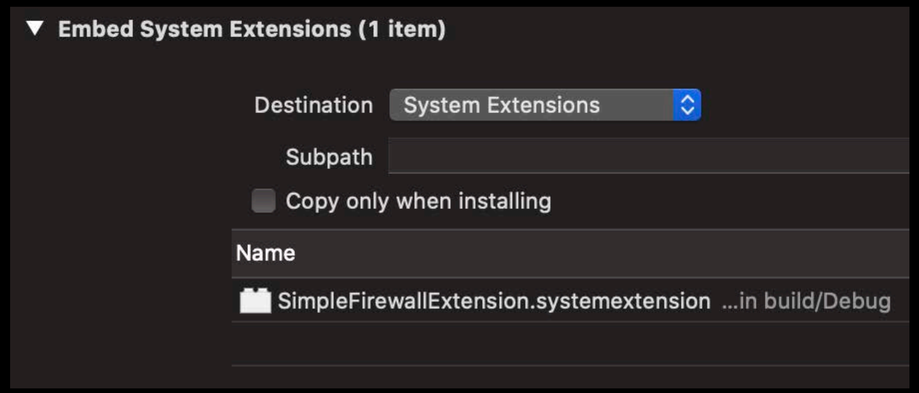
Build extension as another target in your project

Xcode has templates for

- Network Extension
- DriverKit Driver

Copy the extension into app bundle

- Copy Files Phase in app target
- `Contents/Library/SystemExtensions`



Code Signing

Code Signing

Sign System Extension with your App’s signing certificate

Code Signing

Sign System Extension with your App’s signing certificate

Team ID of Extension and App must match

- Exception for extensions designed to be included in other developers’ Apps
- For example, driver for a widely-used interface chip
- Use entitlement: `com.apple.developer.system-extension.redistributable`

Code Signing

Sign System Extension with your App’s signing certificate

Team ID of Extension and App must match

- Exception for extensions designed to be included in other developers’ Apps
- For example, driver for a widely-used interface chip
- Use entitlement: `com.apple.developer.system-extension.redistributable`

System Extensions signed with Developer ID must be Notarized

Code Signing

Sign System Extension with your App’s signing certificate

Team ID of Extension and App must match

- Exception for extensions designed to be included in other developers’ Apps
- For example, driver for a widely-used interface chip
- Use entitlement: `com.apple.developer.system-extension.redistributable`

System Extensions signed with Developer ID must be Notarized

Notarization Lab Tuesday, 4:00

Entitlements

Entitlements

Extensions use entitlements to declare their capabilities

- Type of extension
- Type-specific capabilities
- For example, DriverKit device family and transport

Entitlements

Extensions use entitlements to declare their capabilities

- Type of extension
- Type-specific capabilities
- For example, DriverKit device family and transport

Apps containing extensions use `com.apple.developer.system-extension.install`

Entitlements

Extensions use entitlements to declare their capabilities

- Type of extension
- Type-specific capabilities
- For example, DriverKit device family and transport

Apps containing extensions use `com.apple.developer.system-extension.install`

For more information and to request use of entitlements
developer.apple.com/system-extensions/

Entitlements

Extensions use entitlements to declare their capabilities

- Type of extension
- Type-specific capabilities
- For example, DriverKit device family and transport

Apps containing extensions use `com.apple.developer.system-extension.install`

For more information and to request use of entitlements
developer.apple.com/system-extensions/

In Developer Seed, for local development, turn System Integrity Protection off

Installation

Installation

No installer or package necessary — Extensions stay in your app bundle

Installation

No installer or package necessary — Extensions stay in your app bundle

Use the new SystemExtensions framework

- `activationRequest` API to make an extension available
- Approved by system administrator
- Submit an `activationRequest` at app launch

Installation

No installer or package necessary — Extensions stay in your app bundle

Use the new SystemExtensions framework

- `activationRequest` API to make an extension available
- Approved by system administrator
- Submit an `activationRequest` at app launch

Extension lifecycle is managed by the system

- Starts whenever needed
- e.g. Driver Extensions start when a matching device is connected

Update and Uninstallation

Update and Uninstallation

To update an extension, update your app

Update and Uninstallation

To update an extension, update your app

System notices extension’s version has changed

- `activationRequest` delegate call to compare versions
- Old version of extension stops, new version starts

Update and Uninstallation

To update an extension, update your app

System notices extension's version has changed

- `activationRequest` delegate call to compare versions
- Old version of extension stops, new version starts

Moving app to Trash deactivates all its extensions

- There is a `deactivationRequest` API, too

Summary

Summary

System Extensions are the replacement for Kernel Extensions

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable
- More secure

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable
- More secure
- Easier to develop

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable
- More secure
- Easier to develop

DriverKit framework is a modernized update of IOKit

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable
- More secure
- Easier to develop

DriverKit framework is a modernized update of IOKit

Writing and debugging an example USB driver

Summary

System Extensions are the replacement for Kernel Extensions

- More reliable
- More secure
- Easier to develop

DriverKit framework is a modernized update of IOKit

Writing and debugging an example USB driver

Building and shipping a System Extension in your app

More Information

developer.apple.com/wwdc19/702

Core OS Lab	Tuesday, 2:00	Thursday, 9:00
Security Lab	Tuesday, 10:00	Thursday, 2:00
Networking Lab	Friday, 9:00	


WWDC19