Article

Understanding game controller backward compatibility

Learn how macOS brings support for the latest game controllers to software that predates the introduction of the Game Controller framework.

Overview

Many existing games and third-party input libraries use the IOKit framework to interface with hardware game controllers. Because the IOKit APIs facilitate communication with hardware only at a low level, you need to write device-specific code for each game controller your app supports. This is often just for one or two popular models, so someone with a different controller model may become frustrated to discover that your app doesn't recognize their game controller.

macOS 14 introduces a human-interface device (HID) emulation layer for apps that use the IOKit HID APIs to discover and receive input from game controllers. It emulates basic inputs from the full roster of controllers that the Game Controller framework supports. This HID emulation is mostly transparent to your app, and it works with all apps that interface with game controllers using the APIs in IOHIDManager.h. However, your app may need to determine whether an `IOHIDDeviceRef` refers to a HID device that the emulation layer synthesizes, or your app may need to prevent matching on all such devices.

Understand the operation of game controller synthetic HID devices

When you connect a game controller that the Game Controller framework supports to a Mac running macOS 14 or later, the system creates a *synthetic HID device* in the I/O registry. There can only be one synthetic HID device per connected game controller. If the game controller is HID-compatible, the system creates the corresponding synthetic HID device in the I/O registry

alongside the HID device for the game controller. The characteristics of each synthetic HID device match those of a well-supported game controller, the Xbox 360 wired controller.

When your app calls `IOHIDDeviceOpen` on an `IOHIDDeviceRef` that refers to a synthetic HID device in the kernel, the Game Controller framework checks if the emulation layer is enabled. If the emulation layer is enabled, the Game Controller framework begins translating input events from the real game controller into the equivalent input events that an Xbox 360 wired controller produces. If the emulation layer is not enabled, calling `IOHIDManagerCopyDevices` doesn't return any game controller synthetic HID devices, and attempting to directly instantiate an `IOHIDDeviceRef` with an `io_service_t` that refers to a game controller synthetic HID device fails.

The emulation layer is disabled by default. People can enable it for your game in the system game controller settings by adding a game-specific input customization, and enabling the "Increase controller compatibility" switch.

Discover and identify game controller synthetic HID devices

When your app uses the APIs in `IOHIDManager.h` or `IOKitLib.h` to query the HID devices in the I/O registry, game controller synthetic HID devices that match your specified criteria also return. Each game controller synthetic HID device contains a `GCSyntheticDevice` property with a value of `true` in its I/O registry entry property table. Your app can check for the presence of this key to determine whether an `IOHIDDeviceRef` or an `io_service_t` refers to a game controller synthetic HID device.

> **Important**
>
> Checking for the presence of the `GCSyntheticDevice` property is the only supported way to determine whether a HID device is a game controller synthetic HID device.

The Game Controller framework defines the `kIOHIDGCSyntheticDeviceKey` constant. If you can't import the Game Controller framework into your code, you can specify the "`GCSyntheticDevice`" string directly.

```
if ( IOHIDDeviceGetProperty(device, CFSTR(kIOHIDGCSyntheticDeviceKey)) == kCFBoolear
    // This is a synthetic HID device.
}
```

To prevent matching on game controller synthetic HID devices, add the `kIOHIDGCSyntheticDeviceKey` with a value of `kCFBooleanFalse` to the matching criteria your code passes to

IOHIDManagerSetDeviceMatching, IOServiceGetMatchingServices(_:_:_:), or similar APIs.

```
IOHIDManagerRef manager = IOHIDManagerCreate(kCFAllocatorDefault, kIOHIDManagerOptic
IOHIDManagerSetDeviceMatching(manager, (__bridge CFDictionaryRef)@{
    @kIOProviderClassKey: @kIOHIDDeviceKey,
    @kIOHIDGCSyntheticDeviceKey: @(NO)
});
```

See Also

Game Controller framework migration from IOKit

`var kIOHIDGCSyntheticDeviceKey: String`

A key that specifies whether the device is a game controller synthetic HID device.