**©** Developer Discover Design Develop Distribute Account Q Search by keywords or tags **Developer Forums** Handling `shouldFreeInfo` Parameters This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator. There are three Core Foundation routines that have a shouldFreeInfo parameter: **©** 162 • <u>CFMachPortCreate</u> CFMachPortCreateWithPort • CFMessagePortCreateLocal These routines are correctly documented but that documentation fails to cover the subtleties involved. The critical point is that there are three distinct return cases: • NULL , indicates a failure (A) - In this case shouldFreeInfo will always be true. non-NULL , with shouldFreeInfo false (B) — Here the routine has created a new CF object and thus has retained a reference to the context.info non-NULL , with shouldFreeInfo true (C) — Here the routine has returned an existing CF object, which has already retained its context.info parameter, and thus *your* context.info parameter went unused. You should, as the name suggests, free it. Note CFMachPortCreate can never result in case C. On the one hand, this makes sense because it always allocates a new underlying Mach port. On the other hand, it's a little weird that this routine has a shouldFreeInfo parameter at all! In most cases you're calling these routines from Objective-C or Swift, and thus you want to put an object reference into the context.info parameter. The easiest way to do this is to use the retain and release callbacks. MyObject \* myObject = ...; CFMessagePortContext context = {  $\cdot$ version = 0, .info = (\_\_bridge void \*) myObject, .retain = CFRetain, .release = CFRelease CFMessagePortRef port = CFMessagePortCreateLocal( CFSTR("com.example.myPortName"), messagePortCallback, &context, if (port == NULL) { ... failure ... } else { ... success ... Note The Swift version of this code is a little complex because of the required casting (Swift tries to discourage you from doing anything this unsafe). You can find it at the end of this post. This assumes MyObject is defined like so: @implementation MyObject - (NSData \*)handleMessageOnLocalPort:(CFMessagePortRef)local msgID:(SInt32)msgID data:(NSData \*)data { ... message handling code here ... static CFDataRef messagePortCallback(CFMessagePortRef local, SInt32 msgid, CFDataRef data, void \* info) { MyObject \* obj = (\_\_bridge MyObject \*) info; return (\_\_bridge CFDataRef) [obj handleMessageOnLocalPort:local msgID:msgid data:(\_\_bridge NSData \*) data]; If the CF object holds on to the context.info value, it will call the retain routine to do exactly that. Thus, you can deal with success or failure by looking solely at the function result, ignoring shouldFreeInfo . In fact, it's fine to pass NULL to that parameter. However, there are some cases where looking at shouldFreeInfo makes sense: If the object referenced by the context.info parameter has complex state, such that you need to call a method to tidy up that state, then you'll need to look at shouldFreeInfo to determine whether that's necessary. For example, imagine MyObject is responsible for an open file and you need to close that file in order to tidy up. In that case you might have code like this: Boolean shouldFreeInfo; CFMessagePortRef port = CFMessagePortCreateLocal( CFSTR("com.example.myPortName"), messagePortCallback, &context, &shouldFreeInfo if (shouldFreeInfo) { [myObject close]; if (port == NULL) { ... failure ... } else { ... success ... **IMPORTANT** It's safe to reference shouldFreeInfo in all cases because the value is set even if the routine returns NULL • If, within the context of your program, it only makes sense for one object to be handling this task, you should specifically detect case C and fail. Boolean shouldFreeInfo; CFMessagePortRef port = CFMessagePortCreateLocal( NULL, CFSTR("com.example.myPortName"), messagePortCallback, &context, &shouldFreeInfo if ( (port == NULL) || shouldFreeInfo ) { … failure … } else { ... success ... Easy, eh? Share and Enjoy — Quinn "The Eskimo!" Apple Developer Relations, Developer Technical Support, Core OS/Hardware let myEmail = "eskimo" + "1" + "@apple.com" Here's the Swift version of the core CFMessagePortCreateLocal example: let myObject: MyObject = ... var context = CFMessagePortContext( version: 0, info: Unmanaged.passUnretained(myObject).toOpaque(), retain: { infoQ in guard let info = infoQ else { return nil } let obj = Unmanaged<MyObject>.fromOpaque(info) let objResult = obj.retain() return UnsafeRawPointer(objResult.toOpaque()) release: { infoQ in guard let info = infoQ else { return } Unmanaged<MyObject>.fromOpaque(info).release() copyDescription: nil let portQ = CFMessagePortCreateLocal( "com.example.myPortName" as NSString, { (localQ, msgid, data, info) -> Unmanaged<CFData>? in let local = localQ let data = (data as Data?) ?? Data() let obj = Unmanaged<MyObject>.fromOpaque(info!).takeUnretainedValue() guard let result = obj.handleMessage(localPort: local!, msgID: msgid, data: data) else { return nil return Unmanaged.passRetained(result as NSData) &context, nil guard let port = portQ else { ... failure ... ... success ... This looks more complex than it actually is. The main difference between it and the Objective-C version is that CFRetain and CFRelease are not available in Swift, so you have to implement the retain and release callbacks using Unmanaged It assumes this definition of MyObject class MyObject {

func handleMessage(localPort: CFMessagePort, msgID: Int32, data: Data) -> Data? {

... handle the message ...

Agreement.

Reply to this question

```
É Developer
                  Apple Developer Forums
Discover
                              Design
                                                            Develop
                                                                                           Distribute
                                                                                                                         Support
macOS
                              Human Interface Guidelines
                                                             Xcode
                                                                                           Developer Program
                                                                                                                         Articles
iOS
                                                                                           App Store
                                                                                                                         Developer Forums
                              Resources
                                                                                                                         Feedback & Bug Reporting
watchOS
                              Videos
                                                             Swift Playgrounds
                                                                                           App Review
tvOS
                              Apple Design Awards
                                                             TestFlight
                                                                                           Mac Software
                                                                                                                         System Status
Safari and Web
                              Fonts
                                                                                           Apps for Business
                                                             Documentation
                                                                                                                         Contact Us
                                                                                           Safari Extensions
Games
                              Accessibility
                                                             Videos
                                                                                                                          Account
                                                                                           Marketing Resources
Business
                              Internationalization
                                                             Downloads
                                                                                                                         Certificates, Identifiers &
                                                                                           Trademark Licensing
Education
                              Accessories
                                                                                                                         Profiles
WWDC
                                                                                                                         App Store Connect
To view the latest developer news, visit News and Updates.
Copyright © 2021 Apple Inc. All rights reserved. Terms of Use Privacy Policy License Agreements
```

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation