What is the best way to solve an Objective-C namespace collision?

Asked 16 years, 2 months ago Modified 1 year, 11 months ago

Viewed 43k times

178

Objective-C has no namespaces; it's much like C, everything is within one global namespace. Common practice is to prefix classes with initials, e.g. if you are working at IBM, you could prefix them with "IBM"; if you work for Microsoft, you could use "MS"; and so on. Sometimes the initials refer to the project, e.g. Adium prefixes classes with "AI" (as there is no company behind it of that you could take the initials). Apple prefixes classes with NS and says this prefix is reserved for Apple only.

So far so well. But appending 2 to 4 letters to a class name in front is a very, very limited namespace. E.g. MS or AI could have an entirely different meanings (AI could be Artificial Intelligence for example) and some other developer might decide to use them and create an equally named class. *Bang*, namespace collision.

Okay, if this is a collision between one of your own classes and one of an external framework you are using, you can easily change the naming of your class, no big deal. **But what if you use two external frameworks, both frameworks that you don't have the source to**

and that you can't change? Your application links with both of them and you get name conflicts. How would you go about solving these? What is the best way to work around them in such a way that you can still use both classes?

In C you can work around these by not linking directly to the library, instead you load the library at runtime, using dlopen(), then find the symbol you are looking for using dlsym() and assign it to a global symbol (that you can name any way you like) and then access it through this global symbol. E.g. if you have a conflict because some C library has a function named open(), you could define a variable named myOpen and have it point to the open() function of the library, thus when you want to use the system open(), you just use open() and when you want to use the other one, you access it via the myOpen identifier.

Is something similar possible in Objective-C and if not, is there any other clever, tricky solution you can use resolve namespace conflicts? Any ideas?

Update:

Just to clarify this: answers that suggest how to avoid namespace collisions in advance or how to create a better namespace are certainly welcome; however, I will not accept them as **the answer** since they don't solve my problem. I have two libraries and their class names collide. I can't change them; I don't have the source of either one. The collision is already there and tips on how it could have been avoided in advance won't help anymore. I can forward them to the developers of these frameworks and hope they choose a better namespace in the future, but for the time being I'm searching a solution to work with the frameworks right now within a single application. Any solutions to make this possible?

objective-c cocoa macos namespaces

Share

Improve this question

Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 ● 1

asked Oct 7, 2008 at 13:27

Mecki

133k ● 34 ● 259 ● 269

7 You have a good question (what to do if you need two frameworks that have a name collision) but it is buried in the text. Revise to make it clearer, and you'll avoid simplistic answers like the one you have now. – benzado Oct 7, 2008 at 19:53

4 This is my biggest gripe with the current design of the Objective-C language. Look at the answers below; those that actually address the question (NSBundle unloading, using DO, etc) are hideous hacks that just shouldn't be necessary for something as trivial as avoiding a namespace conflict. – erikprice Mar 1, 2009 at 17:30

@erikprice: Amen. I'm learning obj-c, and hit this very issue. Came here looking for a simple solution.... lame.

– Dave Mateer Feb 5, 2010 at 19:36

1 For the record, technically both C and Objective-C provide support for multiple name spaces — not exactly what the OP is looking for, though. See objectivistc.tumblr.com/post/3340816080/… – user557219 Feb 20, 2011 at 9:41

Hmm, I didn't know that. Kind of a terrible design decision no? – Nico May 20, 2012 at 2:44

13 Answers

Sorted by: Highest score (default)

94

Prefixing your classes with a unique prefix is fundamentally the only option but there are several ways to make this less onerous and ugly. There is a long discussion of options here. My favorite is the `@compatibility_alias` Objective-C compiler directive (described here). You can use `@compatibility_alias` to "rename" a class, allowing you to name your class using FQDN or some such prefix:

```
@interface COM_WHATEVER_ClassName : NSObject
@end

@compatibility_alias ClassName COM_WHATEVER_ClassName
// now ClassName is an alias for COM_WHATEVER_ClassNam

@implementation ClassName //OK
//blah
@end

ClassName *myClass; //OK
```

As part of a complete strategy, you could prefix all your classes with a unique prefix such as the FQDN and then create a header with all the `@compatibility_alias` (I would imagine you could auto-generate said header).

The downside of prefixing like this is that you have to enter the true class name (e.g. `COM_WHATEVER_ClassName` above) in anything that needs the class name from a string besides the compiler. Notably, `@compatibility_alias` is a compiler directive, not a runtime function so `NSClassFromString(ClassName)` will fail (return `nil`)--you'll have to use `NSClassFromString(COM_WHATERVER_ClassName)`. You can use `ibtool` via build phase to modify class names in an Interface Builder nib/xib so that you don't have to write the full COM_WHATEVER_... in Interface Builder.

Final caveat: because this is a compiler directive (and an obscure one at that), it may not be portable across compilers. In particular, I don't know if it works with the Clang frontend from the LLVM project, though it should work with LLVM-GCC (LLVM using the GCC frontend).

Share Improve this answer Follow

edited Jan 18, 2023 at 8:14
Aleksandr Medvedev
8,948 ● 2 ● 21 ● 53

answered Oct 7, 2008 at 22:17
Barry Wark
108k ● 24 ● 182 ● 206

4 Upvote for the compatibility_alias, I didn't know about that one! Thanks. But it does not solve my problem. I am not in the control to change any prefixes of either framework I'm using, I only have them in binary form and they collide. What should I do about that? – Mecki Oct 7, 2008 at 22:46

In which file (.h or .m) should the @compatibility_alias line go? – Alex Basson Apr 23, 2010 at 14:19

1 @Alex_Basson @compatibility_alias should probably go in the header so that it is visible wherever the @interface declaration is visible. – Barry Wark Apr 23, 2010 at 16:06

I wonder if can you use @compatibility_alias with protocol names, or typedef definitions, or anything else for that matter? – Ali Aug 23, 2012 at 8:12

Good thoughts here. Could method swizzling be used to prevent NSClassFromString(ClassName) from returning nil for aliased classes? It would probably be hard to find all methods that took a class name. – Dickey Singh Nov 4, 2013 at 6:12

48

✓

+50

If you do not need to use classes from both frameworks at the same time, and you are targeting platforms which support NSBundle unloading (OS X 10.4 or later, no GNUStep support), and performance really isn't an issue for you, I believe that you could load one framework every time you need to use a class from it, and then unload it and load the other one when you need to use the other framework.

My initial idea was to use NSBundle to load one of the frameworks, then copy or rename the classes inside that framework, and then load the other framework. There are two problems with this. First, I couldn't find a function to copy the data pointed to rename or copy a class, and any other classes in that first framework which reference the renamed class would now reference the class from the other framework.

You wouldn't need to copy or rename a class if there were a way to copy the data pointed to by an IMP. You could create a new class and then copy over ivars, methods, properties and categories. Much more work, but it is possible. However, you would still have a problem with the other classes in the framework referencing the wrong class.

EDIT: The fundamental difference between the C and Objective-C runtimes is, as I understand it, when libraries are loaded, the functions in those libraries contain pointers to any symbols they reference, whereas in Objective-C, they contain string representations of the names of thsoe symbols. Thus, in your example, you can use dlsym to get the symbol's address in memory and attach it to another symbol. The other code in the library still works because you're not changing the address of the original symbol. Objective-C uses a lookup table to map class names to addresses, and it's a 1-1 mapping, so you can't have two classes with the same name. Thus, to load both classes, one of them must have their name changed. However, when other classes need to access one of the classes with that name, they will ask the lookup table for its address, and the lookup table will never return the address of the renamed class given the original class's name.

Share Improve this answer Follow

edited Oct 8, 2008 at 5:25

community wiki
2 revs
Michael Buckley

5 I believe bundle *unloading* was unsupported until 10.5 or later. – Quinn Taylor Jun 16, 2009 at 17:45

12

Several people have already shared some tricky and clever code that might help solve the problem. Some of the suggestions may work, but all of them are less than ideal, and some of them are downright nasty to implement. (Sometimes ugly hacks are unavoidable, but I try to avoid them whenever I can.) From a practical standpoint, here are my suggestions.

1. In any case, inform the developers of **both** frameworks of the conflict, and make it clear that their failure to avoid and/or deal with it is causing you real business problems, which could translate into lost business revenue if unresolved. Emphasize that while resolving existing conflicts on a per-class basis is a less intrusive fix, changing their prefix entirely (or using one if they're not currently, and shame on

them!) is the best way to ensure that they won't see the same problem again.

2. If the naming conflicts are limited to a reasonably small set of classes, see if you can work around just those classes, especially if one of the conflicting classes isn't being used by your code, directly or indirectly. If so, see whether the vendor will provide a custom version of the framework that doesn't include the conflicting classes. If not, be frank about the fact that their inflexibility is reducing your ROI from using their framework. Don't feel bad about being pushy within reason — the customer is always right. ;-)

3. If one framework is more "dispensable", you might consider replacing it with another framework (or combination of code), either third-party or homebrew. (The latter is the undesirable worst-case, since it will certainly incur additional business costs, both for development and maintenance.) If you do, inform the vendor of that framework exactly why you decided to not use their framework.

4. If both frameworks are deemed equally indispensable to your application, explore ways to factor out usage of one of them to one or more separate processes, perhaps communicating via DO as Louis Gerbarg suggested. Depending on the degree of communication, this may not be as bad as you might expect. Several programs (including QuickTime, I believe) use this approach to provide more granular security provided by using Seatbelt sandbox profiles in Leopard, such that only a specific

subset of your code is permitted to perform critical or sensitive operations. Performance will be a tradeoff, but may be your only option

I'm guessing that licensing fees, terms, and durations may prevent instant action on any of these points. Hopefully you'll be able to resolve the conflict as soon as possible. Good luck!

Share Improve this answer Follow

answered Jun 16, 2009 at 17:43

Quinn Taylor
44.8k ● 16 ● 115 ● 133

8

This is gross, but you could use distributed objects in order to keep one of the classes only in a subordinate programs address and RPC to it. That will get messy if you are passing a ton of stuff back and forth (and may not be possible if both class are directly manipulating views, etc).

There are other potential solutions, but a lot of them depend on the exact situation. In particular, are you using the modern or legacy runtimes, are you fat or single architecture, 32 or 64 bit, what OS releases are you targeting, are you dynamically linking, statically linking, or do you have a choice, and is it potentially okay to do something that might require maintenance for new software updates.

If you are really desperate, what you could do is:

1. Not link against one of the libraries directly
2. Implement an alternate version of the objc runtime routines that changes the name at load time (checkout the objc4 project, what exactly you need to do depends on a number of the questions I asked above, but it should be possible no matter what the answers are).
3. Use something like mach_override to inject your new implementation
4. Load the new library using normal methods, it will go through the patched linker routine and get its className changed

The above is going to be pretty labor intensive, and if you need to implement it against multiple archs and different runtime versions it will be very unpleasant, but it can definitely be made to work.

Share Improve this answer

Follow

answered Feb 28, 2009 at 0:00

Louis Gerbarg
43.4k ● 8 ● 83 ● 91

4

Have you considered using the runtime functions (/usr/include/objc/runtime.h) to clone one of the conflicting classes to a non-colliding class, and then loading the colliding class framework? (this would require the colliding frameworks to be loaded at different times to work.)

You can inspect the classes ivars, methods (with names and implementation addresses) and names with the runtime, and create your own as well dynamically to have the same ivar layout, methods names/implementation addresses, and only differ by name (to avoid the collision)

Share Improve this answer Follow

answered Dec 7, 2008 at 18:35

xtophyr
184 ● 2 ● 5

3

Desperate situations call for desperate measures. Have you considered hacking the object code (or library file) of one of the libraries, changing the colliding symbol to an alternative name - of the same length but a different spelling (but, recommendation, the same length of name)? Inherently nasty.

It isn't clear if your code is directly calling the two functions with the same name but different implementations or whether the conflict is indirect (nor is it clear whether it makes any difference). However, there's at least an outside chance that renaming would work. It might be an idea, too, to minimize the difference in the spellings, so that if the symbols are in a sorted order in a table, the renaming doesn't move things out of order. Things like binary search get upset if the array they're searching isn't in sorted order as expected.

Share Improve this answer Follow

answered Feb 27, 2009 at 7:49

Jonathan Leffler
752k ● 145 ● 946 ● 1.3k

Changing the library on disk is out of question, because license won't allow this. Changing the symbols in memory would be possible, but I see no way how this could be done (loading the lib to memory, modifying it and then passing it to the dynamic linker... see no method for that). – Mecki Feb 27, 2009 at 15:54

3 OK - time to decide which of the two libraries is less important, and to find a replacement. And explain to the one for which you pay but that you're abandoning that the reason you're changing is because of this collision and they can retain your business by fixing your problem.
– Jonathan Leffler Feb 27, 2009 at 21:11

2

`@compatibility_alias` will be able to solve class namespace conflicts, e.g.

```
@compatibility_alias NewAliasClass OriginalClass;
```

However, **this will not resolve any of the enums, typedefs, or protocol namespace collisions**. Furthermore, it does not play well with `@class` forward decls of the original class. Since most frameworks will come with these non-class things like typedefs, you would likely not be able to fix the namespacing problem with just compatibility_alias.

I looked at [a similar problem to yours](), but I had access to source and was building the frameworks. The best solution I found for this was using `@compatibility_alias` conditionally with #defines to support the enums/typedefs/protocols/etc. You can do this conditionally on the compile unit for the header in question to minimize risk of expanding stuff in the other colliding framework.

Share Improve this answer Follow

answered Dec 26, 2013 at 19:04
Michael Chinen
18.6k ● 5 ● 35 ● 45

1

It seems that the issue is that you can't reference headers files from both systems in the same translation unit (source file). If you create objective-c wrappers around the libraries (making them more usable in the process), and only #include the headers for each library in the implementation of the wrapper classes, that would effectively separate name collisions.

I don't have enough experience with this in objective-c (just getting started), but I believe that is what I would do in C.

Share Improve this answer Follow

answered Mar 4, 2009 at 3:54
chrish
2,422 ● 1 ● 19 ● 32

1 But wouldn't you still run into collisions if you tried to include both wrapper headers in the same file (since they each would need to include their respective framework's headers themselves)? – Wilco Jun 1, 2009 at 15:39

0

Prefixing the files is the simplest solution I am aware of. Cocoadev has a namespace page which is a community effort to avoid namespace collisions. Feel free to add your own to this list, I believe that is what it is for.

http://www.cocoadev.com/index.pl?ChooseYourOwnPrefix

Share Improve this answer Follow

answered Oct 7, 2008 at 14:14

Ryan Townshend
2,394 ● 21 ● 36

How does prefixing the files help if the objects defined within the files cause the issue? I can surely manipulate the h files, but then the linker will not find the objects at all anymore when liking against the framework :-/ – Mecki Oct 7, 2008 at 15:04

I believe, this would rather be a best–pratice than a solution to the initial problem. – Ali Aug 23, 2012 at 8:15

This doesn't at all address the question – Madbreaks Jan 2, 2013 at 18:39

0

If you have a collision, I would suggest you think hard about how you might refactor one of the frameworks out of your application. Having a collision suggests that the two are doing similar things as it is, and you likely could get around using an extra framework simply by refactoring your application. Not only would this solve your namespace problem, but it would make your code more robust, easier to maintain, and more efficient.

Over a more technical solution, if I were in your position this would be my choice.

Share Improve this answer Follow

edited Mar 5, 2009 at 21:20

answered Mar 1, 2009 at 17:18

Allyn
20.4k ● 17 ● 58 ● 68

0

If the collision is only at the static link level then you can choose which library is used to resolve symbols:

```
cc foo.o -ldog bar.o -lcat
```

If `foo.o` and `bar.o` both reference the symbol `rat` then `libdog` will resolve `foo.o`'s `rat` and `libcat` will resolve `bar.o`'s `rat`.

Share Improve this answer Follow

answered May 20, 2012 at 3:27

wcochran
10.8k ● 6 ● 66 ● 77

0

Just a thought.. not tested or proven and could be way of the mark but in have you considered writing an adapter for the class's you use from the simpler of the frameworks.. or at least their interfaces?

If you were to write a wrapper around the simpler of the frameworks (or the one who's interfaces you access the least) would it not be possible to compile that wrapper into a library. Given the library is precompiled and only **its** headers need be distributed, You'd be effectively hiding the underlying framework and would be free to combine it with the second framework with clashing.

I appreciate of course that there are likely to be times when you need to use class's from both frameworks at the same time however, you could provide factories for further class adapters of that framework. On the back of that point I guess you'd need a bit of refactoring to extract out the interfaces you are using from both frameworks which should provide a nice starting point for you to build your wrapper.

You could build upon the library as you and when you need further functionality from the wrapped library, and simply recompile when you it changes.

Again, in no way proven but felt like adding a perspective. hope it helps :)

Share Improve this answer Follow

answered May 31, 2013 at 16:01

mark
36 ● 2

-1

If you have two frameworks that have the same function name, you could try dynamically loading the frameworks. It'll be inelegant, but possible. How to do it with Objective-C classes, I don't know. I'm guessing the `NSBundle` class will have methods that'll load a specific class.

Share Improve this answer
Follow

answered Jan 18, 2013 at 0:54
MaddTheSane
3,031 ● 25 ● 27