

Q: Can I integrate C++ code into my InterfaceBuilder/Objective-C application? How?

A: Yes, in Release 2 and following releases you can, and it's pretty easy (once you know how)! The procedure breaks down into three categories of things that you must do: compiling, ProjectBuilder and getting the two languages to talk to each other.

### **Compiling**

First, you must use the C++ compiler for *all* of your source files—including the Objective-C sources. To do this, add the following line to your Makefile.preamble:

```
CC=cc++
```

Now that you are using the C++ compiler, you have to notify the compiler when/if your header files

contain non-C++ code. For Objective-C header files, encapsulate your `#import` directives like this:

```
extern "Objective-C"
{
#import <appkit/Application.h>
#import <appkit/Panel.h>
#import <appkit/TextField.h>
#import <appkit/Button.h>
}
```

For regular C header files, encapsulate your `#import` directives like this:

```
extern "C"
{
```

```
#import <appkit/publicWraps.h>
#import <objc/error.h>
#import <objc/NXStringTable.h>
#import <strings.h>
}
```

The C++ "linkage" directive serves two purposes (when importing interface files that contain straight ANSI-C/Objective-C code). It:

- allows you to link with libraries that have not been compiled with the C++ compiler. Since libraries in NEXTSTEP are compiled with the Objective-C compiler (cc, not cc++), you must use the C++ linkage directive when importing interface files that represent NeXT libraries (or any library that is not compiled with cc++).

- tells the compiler to ignore C++ keywords that result in syntax errors when importing ANSI-C or Objective-C interface files. The linkage directive essentially tells the C++ compiler to treat keywords (such as the method names "new", "delete", etc.) as normal identifiers.

## ProjectBuilder

Within InterfaceBuilder you need to add the C++ **.c** and **.h** files to your project. Add the files separately. The **.c** file goes in the Other Sources directory, and the **.h** file goes in the Headers directory.

If you already have a **\_main.m** file, make sure that the option in ProjectBuilder for generating the main file is turned *off*. Then, remove the void declaration of the main procedure by replacing:

```
void main(int argc, char *argv[]) {
```

with:

```
main(int argc, char *argv[]) {
```

## **Modifying Source Code**

Since the nib files generated by InterfaceBuilder are based the AppKit, and it generates source templates in Objective-C, we must envision our program such that Objective-C and nib files are the foundation of our program, and the C++ code is a supporting library.

Now that we can compile, we need to get an Objective-C object and a C++ object to pass messages to one another. Suppose that we have two created objects—a C++ object and an Objective-C object. This is how you would refer to the C++ object and tell it to "do something":

```
class CalcEngine    *cplus_object;  
cplus_object = new CalcEngine;  
cplus_object->doSomething();
```

C++ objects are implemented as regular C structures, so to access public instance variables, or public methods of a C++ object, you dereference the object with the -> syntax as you would a structure member. And this is how you would refer to an Objective-C object from C++:

```
id objectiveObj;  
  
objectiveObj = [ObjectiveObjCls new];  
[objectiveObj doSomethingElse:what];
```

Basically, in either case you use the language constructs of the object to which you are referring, and

embed them in the source file of the other language.

### **Example**

There is an example located in `/NextDeveloper/Examples/AppKit/CalculatorLab++` which illustrates the integration of InterfaceBuilder nib files, Objective-C source code, and C++ source code into one program.

QA584

Valid for 2.0, 3.0

See also `../NEXTSTEP_Developer/Objective_C/NeXT_position_on_C++`.