Q: How can I use a class from a subproject in another project?

A: You can drag any .h file from a subproject into the classes suitcase of another project's nib.   From there you can instantiate, and make connections freely.

In **Release 3**, you can add the .h file into the ProjectBuilder Classes directory by using the Add command in the Files menu. From there, you have to use InterfaceBuilder to instantiate an object, and make the connections.

Q: How can I make connections between objects in my subproject and those in the main project?

A: You can't.   You need to make the association in your code.   The object that is the nib file's owner should take care of communicating with any other objects (including the owners of other nib files).

See ../NEXTSTEP_Developer/AppKit/connecting_between_nibs.rtf for more information regarding object communication between nib files.

Q: When I modify the definition of a class that is located in a subproject, how do I get InterfaceBuilder to recognize the new outlets or actions?   Parse just gives an error: "Cannot find class definition."

A: This is a limitation of InterfaceBuilder in its current incarnation.

In **Release 2**, once a class definition (or .h file) has been dropped into the Classes suitcase from the Workspace Manager, InterfaceBuilder does not have any idea where the files for that class are located. Therefore, you cannot parse classes from a subproject. You must simply drag the .h file onto the Classes suitcase again. InterfaceBuilder then displays an alert panel indicating that there are different outlets or actions for that class. Choose "Replace" and the class changes to reflect the changes made to the .h file.

In **Release 3**, you need to explicitly reparse the .h file by selecting the Classes suitcase, then the proper object in the class browser, and choosing the Parse command from the Operations pop-up list.

Q: The debugger can't find the source files to my subproject classes. Help!

A: In **Release 2**, you must notify **gdb** where the subproject's files are located. You can accomplish this with the **directory** or **idirectory** commands in **gdb**:

```
(gdb) dir <pathname>
```

If you don't want to type this every time you invoke **gdb**, place this command in your **.gdbinit** file. See the **gdb** man page for more information.

In **Release 3**, the **idirectory** command no longer exists. ProjectBuilder generates a **PB.gdbinit** file at the top level project directory. This file contains **"directory"** commands for each subproject. When running the debugger from ProjectBuilder, this file is read in automatically. If you are running gdb from the command line, you can read this file in at startup time by typing the following in a terminal shell:

```
localhost> gdb Foo.app/Foo -x PB.gdbinit
```

Or if you have already run gdb, you can read it in by typing on gdb's command line:

```
(gdb) source PB.gdbinit
```

Note: The following two questions and answer pairs apply to NEXTSTEP   Release 2 only.

Q: Is there an easy way to include subproject class definitions in the main project of my application?

A: Yes, there is.   When you compile the subproject **foo.subproj**, a .h file for the entire subproject is created in the main project's directory.   You can import this .h file.   The name of the .h file, in the case of our example, is **foo.h**.   This file imports all of the class definitions contained in the subproject **foo.subproj**. This may be overkill, especially if you have many classes in your subproject that are not interdependent.   If so, you can also include them individually by using the following import command:

```
#import "foo.subproj/FooBar.h
```

Q: When I try to **make** an application containing a subproject, I get the following error:

```
(cd foo.subproj; make ofiles "NAME=foo" "_CFLAGS= -O -g -Wall"  "OFILE_DIR = obj" "LIBS=")
mkdirs obj
```

```
ld -r -o ../obj/foo.o
ld: no object files specified
```

A: This is an InterfaceBuilder bug that you can encounter if your subproject contains no class definition files. Consider whether you really need a subproject in this case. The workaround is to create a dummy class and add its .h and .m files to your subproject.

QA792

Not valid for 1.0 (subprojects were added for 2.0)
Valid for 2.0, 3.0, 3.1