OPENSTEP

Title: OPENSTEP/NT Developer FAQ
Entry Number: 2462
Creation Date: September 24, 1996
Last Updated: <<DATE>>

Keywords: NT, framework, dll, webobjects

NOTE: Unless otherwise stated, this information applies to NT 4.0 Gold and OPENSTEP/NT 4.1 only. The information presented here may or may not apply to versions of NT before 4.0 gold, or OPENSTEP/NT 4.0.

NOTE: Unless otherwise stated, this information also applies to WebObjects Enterprise.

Q: Why does my application say "Class not loaded" at run time even though I clearly have the framework linked into my project?

A:

Even though it looks like your app build properly, the linker will not actually link in a dll unless you make direct reference to a symbol in the dll. Here is an example of how to directly reference symbols, taken from the template

EOF application:

```
#ifdef WIN32
#import <EOControl/EOControl.h>
#import <EOAccess/EOAccess.h>
#import <EOInterface/EOInterface.h>

void _referenceAllEOFrameworks()
{
      static id a,b,c;
      a = [EODisplayGroup new];    // EOInterface
      b = [EOEntity new];                // EOAccess
      c = [EOEditingContext new];    // EOControl
}
#endif
```

You never need to call this code. Simplying having it in your project is sufficient.

Q: Why do I get error LNK 1120: unresolved externals even though I clearly have the framework linked into my project?

A:

There is a bug in the OPENSTEP/NT 4.1 makefiles that will not export symbols from subprojects within a framework. You can add the following to the Makefile.postamble in your main project as a workaround.

```
ifeq "WINDOWS" "$(OS)"
$(WINDOWS_DEF_FILE): $(OFILES) $(OTHER_OFILES)
        $(SILENT) if [ ! -r "./$(WINDOWS_DEF_FILE)" ] ; then \
                $(ECHO) -n "Generating $(notdir $@)...." ; \
                (cd $(OFILE_DIR); $(DUMP_SYMBOLS) $(OFILES) $(OTHER_OFILES) `$(CAT) $(NAME).ofileList` | $
(EGREP) "(SECT).*(External).*(\|)" | $(EGREP) -v "(__GLOBAL_$I)|(_OBJC_)" | $(SED) "s/ _/ /" > $(TMPFILE)) ; \
```

```
        $(ECHO) "LIBRARY $(NAME).dll" > $(GENERATED_DEF_FILE) ; \
        $(ECHO) "EXPORTS" >> $(GENERATED_DEF_FILE) ; \
        $(CAT) $(TMPFILE) | $(EGREP) "(SECT2)|(.objc_c)" | $(AWK) '{printf "\t%s CONSTANT\n", $$NF}' >> $
(GENERATED_DEF_FILE) ; \
        $(CAT) $(TMPFILE) | $(EGREP) -v "(SECT2)|(.objc_c)" | $(AWK) '{printf "\t%s\n", $NF}' >> $
(GENERATED_DEF_FILE) ; \
        $(RM) $(TMPFILE) ; \
        $(ECHO) "done"; \
    fi
endif
```

Also, globals and functions are not automatically exported from a framework. See below for the workaround.

Q: Why will a project build sometimes fail for seemingly no reason in the link stages?

A: Because you may have the executable running or open in gdb

You may sometimes get errors that look like the following when trying to build your project under NT:

```
D:/NeXT//NextDeveloper/Libraries/libNSWinMain.a
LINK : fatal error LNK1104: cannot open file "C:/Projects/ImageCrop/ImageCrop.debug/ImageCrop.exe"
gcc: Internal compiler error: program ld got fatal signal 127
make[1]: *** [C:/Projects/ImageCrop/ImageCrop.debug/ImageCrop.exe] Error 1
make: *** [debug] Error 2
```

This is likely caused by the fact that the application is either still running, or you still have a debug session running. On NT, it is impossible to delete or overwrite a file while some process is using it.

Q: Why do I get a segmentation fault when trying to use a bundle?

A:

There is a problem with code in a bundle trying to access global variables in the main application. Typically, when the bundle code tries to access the global variable, you will get a segmentation fault. The solution is to put these global symbols into a framework and link both the application and the bundle against the framework. You must also implement additional declarations (see below) to export symbols other than class and category names from your frameworks.

Q: How do I export symbols other than class names from my framework?

A:

The makefiles that ship with OPENSTEP/NT will build a framework properly with class and category names exported, but functions and public variables are not exported. You must take special measures to export them explicitly.

You can create a header file like the one below and include it in your framework, to be both included by the framework and applications that link against the framework.

```
#ifdef WIN32

#ifndef _BUILDING_MY_FRAMEWORK
#define _MY_FRAMEWORK_GOOP     __declspec(dllimport)
```

```
#else
#define _MY_FRAMEWORK_GOOP      __declspec(dllexport)
#endif _BUILDING_MY_FRAMEWORK

#ifdef __cplusplus
#define MY_FRAMEWORK_EXTERN    _MY_FRAMEWORK_GOOP extern "C"
#define MY_FRAMEWORK_EXTERN    extern "C"
#else
#define MY_FRAMEWORK_EXTERN _MO_WINDOWS_DLL_GOOP extern
#define MY_FRAMEWORK_PRIVATE_EXTERN extern
#endif __cplusplus

#endif WIN32
```

You could then create public header files in your framework that use the macros in this header file to export symbols. For instance:

```
#ifdef WIN32
#import "MyFrameworkGoop.h"

// This function is public
MY_FRAMEWORK_EXTERN somePublicFunction();
// This function is private to the framework
MY_FRAMEWORK_PRIVATE_EXTERN int somePrivateVariable;

#endif WIN32
```

Q: Why can't I debug bundles and frameworks?

A: gdb cannot read symbols from relocated bundles and frameworks under NT.

Both bundles and frameworks are dynamically loaded code, and may be relocated in memory if there is a conflict with previous dynamic loaded code. Typically, you will be able to debug the first bundle and framework loaded, but not subsequent ones. You can avoid relocation of dynamically loaded code by specifying a base address to the linker so that your code will be loaded in a non-conflicting range of memory.

Add the following lines to your Makefile.preamble.

For frameworks, use:
WINDOWS_PB_LDFLAGS = -image_base 0x00000000 (fill in an address)

For Bundles, use:
WINDOWS_PB_LDFLAGS = -Xlinker "/BASE:0x00000000" (fill in an address)

A good address to use would be the ones that you see the code being relocated to as things load in gdb.