3.3 Release Notes: Project Builder

This file contains release notes for both the 3.2, 3.1, and 3.0 releases of Project Builder. Items specific to the 3.2 release are listed first, and the Release 3.1 and 3.0 notes follow.

Notes Specific to Release 3.3

Bugs Fixed in Release 3.3

Reference: 40637

Problem: Bundle extensions set in Project Builder don't get written to the **Makefile** on a PA-RISC system.

Description: Bundle Extensions are not written to the **Makefile** from the

attributes section of Project Builder. The BUNDLE_EXTENSION entry in **Makefile** for Project Builder never exists on PA-RISC systems.

Notes Specific to Release 3.2

New Features

The following new features have been added to Project Builder since Release 3.1.

- **Finder mode.**
  A new mode, Finder, has been added to the project window. It supports searching a project for a string. When the string found, the results appear in the area at the bottom of the window. Clicking a result causes Edit to open the file and highlight the string. The ªOptionsº button brings up a panel that allows you to limit or expand the search.

- **Tool project type.**
  A new project type has been added. Tools are executables, but they are **not** applications. Resources are not supported, and no main file is generated. They are intended to be used in two ways. A stand-alone executable, that the user will execute, can be created using the ªNew projectº command. A tool can also be embedded in an application or bundle by creating a tool subproject using the ªNew subprojectº command.

Bugs Fixed in Release 3.2

These bugs have been fixed in Release 3.2:

Reference 29714

Problem Interface Builder and Project Builder don't pay attention to umask.

Description Project Builder, Interface Builder, and Edit all ignore the user's default permissions when creating files. All files are readable by all, and writable by only the user.

Reference: 34465

Problem: The <Default> target isn't provided as a build target.

Description: The <Default> target isn't in the build panel target pop-up list. <None> is there instead, but its use causes the target ª<None>º to be built instead of the default target.

Notes Specific to Release 3.1

New Features

The following new features have been added to Project Builder since Release 3.0.

- **Support for multiple architecture builds (fat files).**
  Project Builder has been enhanced to allow the developer to build applications that will run on both the Intel and Motorola versions of NEXTSTEP. In Builder mode there is now a button labeled ªOptions...º. This brings up a panel which can be used to add arguments, set the remote host, and the architecture(s) that the resulting application will be able to run on. By default, applications are built to run on the type of machine that Project Builder is running on, but this default can be changed using the Preferences panel.

- **Sounds.**
  Project Builder will now (optionally) play different sounds when a build succeeds and when one fails. These can be specified in the preferences panel.

- **User settable tools.**
  Project Builder can now be configured to use different tools. The editor, debugger, and builder can all be set from the preferences panel. Note: The editor selected here will only be used when clicking on error messages. Opening files from the Files mode will still use the system default editor for the file type being opened.

- **Makefile preamble and postamble templates.**
  Project Builder now installs Makefile preambles and postambles by default. These files can be modified to supplement the building of the project. They can also be found in **/NextDeveloper/Makefiles/app** to be copied and used in an

existing project.

- **Target PopUpList.**
  The build panel now contains a popuplist which allows the user to choose the target to be built. Additional targets can be added by selecting ªAdd...º from the list. These targets are **not** saved across invocations. To add permanent targets to the popup use the Preferences panel.

Bugs Fixed in Release 3.1

These bugs have been fixed in Release 3.1:

Reference 29595

Problem Project Builder can create a faulty makefile if several libraries are specified.

Description If many libraries are specified in Project Builder, the LIBS line in the resulting makefile can contain an extra space. Here is an example of the LIBS line generated in such a case:

```
LIBS = -lcommon_g -lipc -lMedia_s -lNeXT_s -l\
 MallocDebug
```

The space between the ª-lº and ªMallocDebugº will cause an error at link time.

Workaround Manually edit the makefile to eliminate the extra space.

Reference: 29346

Problem: If there are no libraries, then the Project Builder generated Makefile dies

Description: To reproduce:

1. Create a project
2. Go to Libraries
3. Remove NeXT_s and Media_s with Command-r
4. Save
5. Build

make gives you the following:

Make: Infinitely recursive macro? Check for substitutions like
LIBS = -lNeXT_s -lsys_s $(LIBS)
Stop.

This is because of following generated Makefile source

LIBS = DEBUG_LIBS = $(LIBS)
PROF_LIBS = $(LIBS)

Reference: 28391

Problem: Once set, the application icon can't be changed to default icon.

Description: In Project Manager, if you set the application icon, there is no way (short of

editing the makefile and **PB.project**) to change the icon back to the default one.

Known Problems

These new bugs have appeared since Release 3.0.

- The <Default> target is not in the build panel target popup. <None> is there instead, but its use causes the target ª<None>º to be built instead of the default target. A workaround is to add the <Default> target using the target add panel. *(Fixed in 3.2)*

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of Project Builder.

Release 3.0 Release Notes:

Project Builder

Project Builder is a new tool for release 3.0. It takes over the project maintenance role from InterfaceBuilder, and adds quite a bit of new functionality. Some of its

features are described below.

- **Converting Old Projects.**
  To convert an existing IB.proj, simply open it, either by double clicking on it in Workspace or by using Project Builder(PB)'s open panel. When you save the resulting project, it will be saved as a ªPB.projectº file in the same directory. Open this file from now to work with the project.

- **Creating New Projects**
  To create a new project, use the new command in the Project Menu. You will be given the choice of creating an application, bundle, or InterfaceBuilder palette. First select a directory. If you want the project to be created in that directory, just hit return. A project named ªPB.projectº will be created there. If you type anything other than ªPB.projectº a new directory will be created with that name, with a PB.project in it. All necessary files for the project type you selected will also be created.

- **The Project Window**
  The project window has three modes. They are: Attributes, Files, and Builder. The Attributes mode lets you set attributes on your project. The Files mode allows you to add, remove, or open the project's files. The Builder mode allows you to build the project.

- **Adding Files to the Project**
  There are several ways to add a file to a project. The file **must** exist first. PB

does not create files for you, except when you are creating a brand new project. The file can either be in the project directory itself, or somewhere else. To add it do one of the following.

1) Drag it from workspace into the project window (in Files mode). If you drag it to the suitcase it belongs in, that suitcase will open up. If you let it go, it will be added to that suitcase. If instead you drag it to the project suitcase, the suitcase will open up as the suitcase that the file will be added to (if PB can figure it out). The Classes suitcase takes .m files, the Headers takes .h files, and so on. *Other Sources* refers to files that are not headers or classes, but need to be compiled and linked into the target of the project (application, bundle or palette). *Other Resources* refers to files that need to be copied into the target. *Supporting Files* refers to files that are necessary to maintain the project, but don't end up in the target.

2) You can select a suitcase and use the ªAdd...º command from the ªFilesº menu. You will get a panel allowing you to add a file.

3) You can use the Service that PB supplies to other applications. Some apps will have an entry called ªProjectº in their Services menu. Off of this there will be two commands, ªAdd toº and ªBuildº. The ªAdd toº command will add the currently selected file to the project that is in the same directory.

· **Building the Project**

Building the project actually runs a ªmakeº in the directory that the project is in.

Status is indicated, and errors are reported.

There are several ways to build a project. You can click on the ªRunº button to build and run an application. You can click on the ªDebugº button to build and debug an application. In ªBuilderº mode, you can click on the ªBuildº button to build the target. Builder mode gives you two other fields. The ªArgsº field will be passed through as an argument to the make that PB starts. The ªHostº field allows you to perform the make on a different host.

Once the build is running, any syntax errors will appear. You can click on them to bring up Edit at the line the error occurred at.

· **Shortcuts/Tips**

Below is a list of features that may not be obvious at first glance.

1) Alt-clicking on the ªRunº button will run the application **without** building it first. This also works with the ªDebugº button.

2) Control-dragging in a files list allows you to reorder the the files. This can be especially important in dealing with libraries, as it specifies the link order.

3) Alt-double-clicking on the icon of a selected file *selects* that file in Workspace, instead of opening it.

4) Double-clicking on a suitcase, or its entry in the browser is equivalent to using the

ªFiles->Addº menu command with that suitcase selected.

5) Command-double-clicking on a source file opens both the file and its associated header file, if it exists.