3.3 Release Notes: Interface Builder

This file contains release notes for the 3.3, 3.2, 3.1, and 3.0 releases of Interface Builder.

Notes Specific to Release 3.3

There are substantial changes for Release 3.3. Interface Builder for Release 3.3 is described in the new manual *Working With Interface Builder* available in hard copy and on-line in **/NextLibrary/Documentation/NextDev/DevTools/NewInterfaceBuilder**.

Notes Specific to Release 3.2

There are no changes to note for Release 3.2.

Notes Specific to Release 3.1

Bugs Fixed in Release 3.1

These bugs have been fixed in Release 3.1:

Reference 28510

Problem Interface Builder can lose target/action connection information for a custom object if the object's class is reassigned.

Description If you reassign the class of a custom object to a subclass of its former class, connection information can be lost. For example, if you reassign the class of a custom View from GraphView to ColorGraphView (a subclass of GraphView), a Button whose target had previously been the GraphView object becomes disconnected.

Reference: 29773

Problem: Invoking **registerDocumentController:** during Interface Builder's launch fails.

Description: If a custom palette is loaded as part of Interface Builder's launch and some object within the palette attempts to register itself as a document controller, the registration fails. This is because the List object that would store the document controller isn't created until after the Palettes window appears.

Reference: 30400

Problem: Resizing Views is slow.

Description: Resizing Views within a window of an application under construction is very slow. The larger the View, the slower the resizing.

Known Problems

These new bugs have appeared since Release 3.0.

Reference: 31717

Problem: Interface Builder crashes if you try to parse multiple header files at the same time.

Description: In the Classes display of the File window, choose Parse from the Operations pop-up list. If you choose more than one header file in the Open panel that appears and click OK, Interface Builder crashes.

Workaround: Parse only one header file at a time.

Reference: 33245

Problem: If you delete all the menu items from an application's main menu, you can no longer add new items.

Description: After deleting the last menu item from an application's main menu, you can't add any new ones. Then, if you close the main menu window, double-clicking its icon in the File window fails to bring it back.

Workaround: Don't delete the last menu item before adding new ones.

Reference: 32990

Problem: Library sounds aren't accepted by the Sounds suitcase.

Description: Few of the sounds found in the well-known locations (**~/Library/Sounds**, **/LocalLibrary/Sounds**, **/NextLibrary/Sounds**) show up in Interface Builder's Sounds suitcase. If you drag a sound from one of these locations into Interface Builder's File window, it won't be accepted. Instead, Interface Builder displays a panel saying that the sound already exists.

Workaround: If you're trying to assign one of these sounds to a button, enter the name of the sound in the Sound field of the Button inspector.

Notes Specific to Release 3.0

These notes were included with the Release 3.0 version of Interface Builder. Sections that are no longer relevant have been marked with an italicized comment.

Interface Builder has been fully rewritten for release 3.0. Here are the highlights:

- Project management has been removed from Interface Builder. It has been moved to ProjectBuilder, which is installed in **/NextDeveloper/Apps**. Project Builder creates projects for you, creating default project files (a makefile, a main file, and so on), and lets you access other development tools such as Edit and Interface Builder.

  Given this new arrangement, it's best to start a new application by creating a project in Project Builder, which provides you with a default nib file for the type of project you are creating. You can then open this nib file and make the changes appropriate for your application.

- The nib file format has changed. Nib files are no longer files but file packages.

  Important: Nib files created with the new Interface Builder can no longer be included in Mach-0 segments of an executable file.

- The format for loadable palettes has changed. The old format is no longer supported.

- Most inspectors don't have OK or Revert buttons.

- The File window has changed. It now has a shelf with suitcases corresponding the object, image, sound, and class resources in your application.

- Interface Builder now supports the NEXTSTEP help system. Once a Help directory has been added to your project using Project Builder, Interface Builder's Help Builder panel will let you view help files and connect user-interface objects to specific pieces of that text.

- Inspector categories have been changed. They now consist of Attributes, Connections, Size, and Help.

The process of making custom palettes has changed for the 3.0 release. In addition, you can now make custom palettes for non-View objectsÐWindows, Panels, MenuCellsÐas well as for objects that are instantiated by being dragged into the File window. The tutorial in **/NextLibrary/Documentation/NextDev/DevTools/18_CustomPalette** has been updated to describe the new process for creating palettes of View objects. For more information on creating other types of palettes, see the outlines below and refer to **/NextLibrary/Documentation/NextDev/GeneralRef/08_InterfaceBuilder**.

How to Make a Palette of Views

- Open ProjectBuilder
- Choose the Project/New... command
- in the SavePanel set the name of your palette project to 'MyPalette' for instance.
- in the SavePanel use the popup to set the ProjectType to 'Palette'.
- hit OK
- a new palette project has been created with one nib file (MyPalette.nib) and one class file (MyPalette.[hm])
- add your extra files in the project
- open MyPalette.nib and add the views you want in the palette
- open the ascii file palette.table:

```
Class = MyPalette;                  /* (a subclass of IBPalette) */
NibFile = MyPalette;                /* (a nib file name) */
/* Icon =;                          (a tiff/eps file name) */
/* ExportClasses = ();              (a list of class names) */
/* ExportIcons = ();                (a list of icon names) */
/* ExportSound = ();                (a list of sound names) */
```

- to get a custom icon for your palette, add a tiff file MyPaletteIcon.tiff in your project images (it should be smaller than 48x48 and have appropriate alpha in order to fit in a 48x48 bordered button). Then change the 3rd line of palette.table to:

```
...
Icon = MyPaletteIcon;
...
```

- if you want to have some custom classes (resp. images, resp. sounds) that you added to your palette nib file, to be accessible to the palette users, edit the appropriate lines in palette.table:

```
...
ExportClasses = (MyFirstClass, MySecondClass);
...
```

- build

How to Make a Palette of Objects/Menus/Windows

- Proceed as above
- The second step is now to associate a view in the palette view to some object/menucell/window. If you want to do it for a window for instance
- Create another window in MyPalette.nib
- Create 2 new instance variables for the class MyPalette 'windowView' and 'windowData' for instance
- Connect windowView to the View in the palette that will represent the window (that the user will drag and drop
- Connect windowData to the new window you created
- Implement the following method for MyPalette

```
- finishInstantiate {
[self associateObject:windowData type:IBWindowPboardType with:windowView];
return self;
}
```

- For objects (that the user will deposit in the file window) use the IBObjectPboardType constant, for MenuCells without submenu use IBMenuCellPboardType, for MenuCells with submenu use IBMenuPboardType

How to Make an Inspector

- implement your own IBInspector subclass:

```
#import <apps/InterfaceBuilder.h>
@interface MyInspector:IBInspector <IBInspectors> {
....
}
@end
```

```
@implementation MyInspector
- (BOOL)wantsButtons { return YES/NO; }
- revert:sender {
      ....
      return [super revert:sender];
}
- ok:sender {
      ....
      return [super ok:sender];
}
@end
```

- define getInspectorClassName for your own class:

```
- (const char *)getInspectorClassName { return "MyInspector"; }
```

Known Problems

- InterfaceBuilder defaults are not written by the standard mechanism. If you get any problems with the defaults (for example, which palettes are loaded at startup) and want to get rid of them just remove the file **~/.Next/defaults.nibd**

- Alternate drag to move an object between windows without breaking the connections is not working.

- Copy/paste is not working for the Class Browser.

- Copy/paste is not working for the Class inspector.

- IB crashes after permitting you to open non-nib files if you type the file name in the

OpenPanel rather than selecting a name.

- IB is unpredictable when going into Test mode if there is a target/action connection to a custom subclass of Control.

- Cannot connect (or select) a custom view that is grouped within a scrollview. You'll need to use Ungroup to make it accessible.

- If you change the class of an object to a subclass of itself, then any control that was connected to it gets disconnected. *This is fixed in 3.1 (ref. bug 28510).*

- Unable to create matrix with custom subclass of control.

- Accidently hitting Cmd-r (or command key equivalents other than cmd-c, cmd-x, cmd-v) while saving causes a crash.