Interface Builder Guide

Technical Publications

CHAPTER 1

Introduction

This chapter will discuss the basic windows in Interface Builder and some of the basic concepts you need to understand. Right now, it contains

Nib File Window

This section describes what’s in the nib file window: the main window of Interface Builder.

“Instances Display of the Nib File Window” (page 1)

“Icon Mode” (page 2)

“Outline Mode” (page 2)

“Expanding Objects in Outline Mode” (page 3)

“Identifying Objects in Outline Mode” (page 3)

“Classes Display of the Nib File Window” (page 4)

Instances Display of the Nib File Window

The Instances display of the nib file window displays the objects in your nib file. There are two ways to view it: icon mod and outline mode. To switch between the modes, use the buttons on the upper-left corner of the nib file window. To see the icon mode, press ■. To see the outline mode, press ☰.

Icon Mode

The icon mode the Instances display shows objects as icons. This mode doesn't show all objects, just the top-level objects—those objects that are not contained by another object. Windows and panels and most controller objects (that is, objects that manage an application or a window) are top-level objects. Although these objects may contain other objects (for instance, a window contains one or more views), no other object contains them.

The graphical representation of objects in icon mode makes it an ideal interface for many operations. Use it with the objects in your interface to do simple operations like making connections.

Outline Mode

The outline mode shows the objects in the nib file in full detail, including their connections with each other.

The most important advantage of the outline mode is that it shows all objects in the nib file, not merely the top-level objects. It also shows all connections: both connections into an object and connections from an object to other objects.

The outline mode starts by listing the top-level objects in the nib file. By clicking the small gray circular button next to an instance, you can see what other objects it contains. Click the triangle button to see what connections go into or out of an object.

You can connect objects in outline mode, so there's no need to drag a connection line into a possibly crowded interface. From the outline mode, you can also find where an objects appears in the interface and disconnect objects.

Objects in outline mode are identified first by class name and then by title (if the object has one). If the title is obscured, you can resize the nib file window until it is visible.!

Expanding Objects in Outline Mode

In outline mode, objects that contain other objects have a small circle to their left that's filled with gray. The subordinate objects are usually subviews of a window, panel, or another view object. You display these contained objects by expanding the container object.The outline mode shows you the view hierarchy, or which objects are visually contained within other objects in the user interface. To see the class hierarchy, or which classes are subclasses of other classes, use a development environment with a class browser.

Click a gray circle to expand a list of the object's component objects; click it again to collapse the list. Expansions may nest many levels. To expand everything within an object, Command-click the gray circle. Collapse the list back to the original level by Command-clicking the gray circle again.

Identifying Objects in Outline Mode

In the outline, you can find where an objects appears in the interface mode two different ways.

To display a representation of an object under the cursor, do this:

1. While holding down the Alternate key, click the object's name.

   A representation of the object appears under the cursor. If the object isn't in the user interface, an icon appears instead. For example, a cube appears for custom objects and a miniature window appears for windows and panels.

   Nothing appears if you click the File's Owner, First Responder, and Main Menu objects.

To display an arrow pointing at the object in the interface, do this:

1. While holding down the Control and Shift keys, click the object's name.

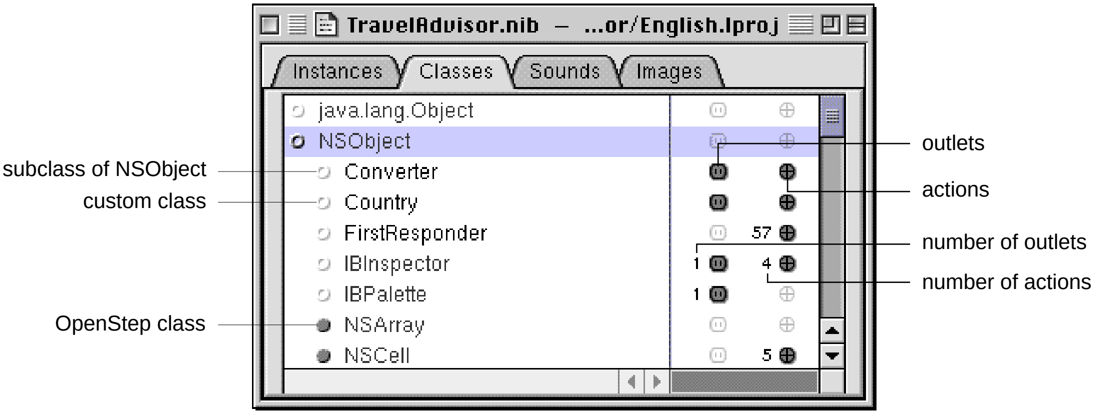
   The window or panel that contains the object comes to the front and a large arrow points at it.

   Nothing appears if you click the File's Owner, First Responder, and Main Menu objects.

Classes Display of the Nib File Window

The Classes display of the nib file window shows the classes that the current nib file knows of and lets you browse through both Yellow Box classes and custom classes. The Classes display can also show you class inheritance and the names of a class's outlets and actions.

Figure 1-1 The classes display



You can tell whether a class is a custom class or Yellow Box class by looking at its name. If the name is black, it's a custom class. If the name is gray, it's a Yellow Box class.

The circle to the class name's left tells you about subclasses. If a class has subclasses, the circle is dark gray. To view the subclasses, click the circle. The circle becomes empty and the subclasses appear indented underneath the class. To hide the subclasses, click the circle again. The circle becomes filled in and the subclasses disappear. If the class has no subclasses, the circle is light gray.

To the right of the class's name are icons for its outlets (⊡) and actions (⊕). The number of outlets and actions appear beside the icons. To view them, click either icon.

Here are some keyboard shortcuts for the Classes display:

- To show a class's subclasses, select it and press the Right Arrow key. To hide the subclasses, press the Left Arrow key.
- To move up and down in the list of classes, press the Up Arrow and the Down Arrow keys.
- To select a class, type the first few letters of its name.

Basic Terms

Outlet

An outlet is an instance variable that points to another object. Objects use outlets to communicate with other objects; they simply send messages to the object identified by the outlet.

In Interface Builder, you can declare and set outlets for the custom objects in your application. You can also set ready-made outlets in many Application Kit objects, such as browsers. Once initialized, the connection information for the outlet is stored in the nib file. At run time, the nib file is unarchived and the outlet is re-initialized with the connection information.

The Application Kit defines two types of outlets that you can use to establish specialized connections with other objects: **delegate**s and **target**s.

delegate

A delegate is an object that acts on behalf of another object. Many Application Kit classes define delegate "Outlet"s as an alternative to subclassing. All your object must do is register itself as a delegate of the Application Kit object. At certain key times, the Application Kit object sends messages to its delegate, giving it an opportunity to participate in processing and sometimes the chance to veto behavior.

For example, browsers ask their delegates to supply the cells for browser columns, and the application informs its delegate when it is initialized, hidden, and activated.

target

A target is a special kind of “Outlet”. It identifies objects that can respond to “action” messages. When a user activates an NSControl object (for instance, clicking a button or moving a slider), that object sends an action message to the target. The action message gives application-specific meaning to the original mouse or key event. For example, you could connect a custom object in your application as the target of a button so that when the button is clicked, your object performs a method that fills all of the text fields in the window with appropriate information.

Like a “delegate”, a target must implement methods to respond to the messages it’s sent. But unlike a delegate, which receives messages chosen from a limited set defined by another object, a target responds to any action message you choose to define.

You can also make one object a target of a second object programmatically by sending setTarget: to the second object.

action

When a user manipulates an NSControl object, the object receives an event message, which it translates into a message that is meaningful within the application. It then send this message to another object. These application-specific messages initiated by an NSControl object are called action messages, and the methods they invoke are called action methods.

NSControl, an abstract class, defines for its many subclasses (such as NSButton, NSScroller, NSTextField, and NSForm) a paradigm for inter-object communication—action messages. But NSControl objects don’t act alone: they always contain one or more NSActionCells (or one of its subclasses). The NSActionCell superclass defines instance variables for the two elements essential to an action message:

- target: the object that’s responsible for responding to the user’s action
- action: the method that specifies what the target is to do

Action methods take a single argument, the id of the NSControl object that sends the message. This argument enables the receiver to ask the control for more information, if it’s needed.

An NSControl can send a different action message to a different target for each NSActionCell it contains. Different NSControls dispatch action messages differently; for instance, an NSButton generally sends action messages on a mouse-up event, but an NSSlider usually sends action messages continuously, as long as the mouse button is pressed.

CHAPTER 2

Setting Attributes

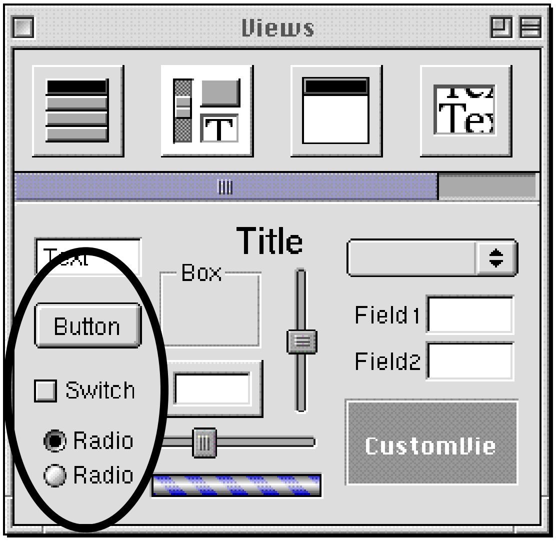
You can customize what the objects in your interface do and look like by setting their attributes. This chapter will describe the attributes you can set for each of the objects on Interface Builder's palettes. For now, it only describes button attributes.

Buttons

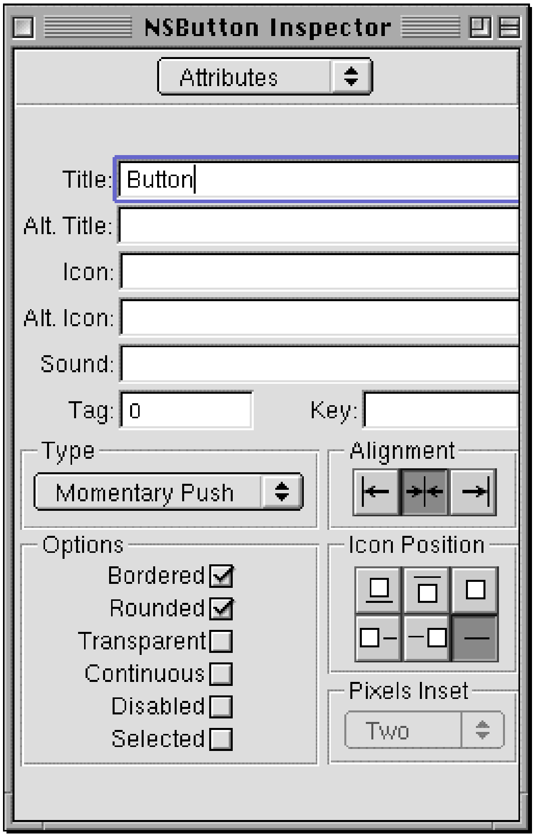
The Attributes display of the NSButton Inspector panel lets you set a button's type, title, icon, alternate title and icon, and other characteristics. The Views palette holds three types of buttons. They are all instances of the class NSButton with different defaults selected in the Inspector.

- Button, a basic push button
- Switch, a check box button
- Radio, an instance of NSMatrix containing radio buttons. The matrix makes sure only one button is selected at a time.

The three button types are circled in the following figure



This is the Inspector for a push button:



NSButton Inspector

Attributes

Title: Button

Alt. Title:

Icon:

Alt. Icon:

Sound:

Tag: 0

Key:

Type: Momentary Push

Alignment

Options

Bordered ☑

Rounded ☑

Transparent ☐

Continuous ☐

Disabled ☐

Selected ☐

Icon Position

Pixels Inset: Two

For more information on a button, see

“What Is a Button?” (page 12)

For more information on setting button attributes, see

“Setting the Button Type” (page 12)

“Setting Button Titles” (page 13)

“Setting Button Sound” (page 13)

“Setting Button Icons” (page 14)

“Setting the Pixel Inset” (page 15)

What Is a Button?

The buttons in the Views palette are essentially two-state NSControl objects. When a user clicks a button, an action message is sent to a target object. It is two-state because it is either on or off, and when it is on, it typically sends its action message. For a button, the states are also known as "normal" (off) and "alternate" (on).An NSButton object can actually have three states—on, off, and mixed—but you cannot enable the third state from Interface Builder. You must add code to your application to do that.

Like most objects on the Views palette, a button is actually a compound object: an NSButton object and an NSButtonCell object. Most of NSButton's methods match identically declared methods in NSButtonCell. Aside from dispatching the action message, NSButton's unique role is to set the font of the key equivalent, and to manage the highlighting or depiction of the NSButton's current state.

Setting the Button Type

The Button Type pop-up menu lets you choose from six types of buttons.

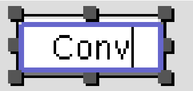
Type	**Description**
Momentary Push	As the button is pressed, it's highlighted and appears to be pressed. (It returns to normal after the user releases the mouse button.) This is the default for a push button.
Momentary Change	As the button is pressed, the alternate button title and icon appear. (It returns to normal after the user releases the mouse button.)
Momentary Light	As the button is pressed, it's highlighted but it does not appear to be pressed. (It returns to normal after the user releases the mouse button.)

Type	Description
Push On/ Push Off	After the button is clicked once, it's highlighted and appears to be pressed. After the button is clicked again, it returns to normal.
On/Off	After the button is clicked once, it's highlighted. After the button is clicked again, it returns to normal.
Toggle	After the button is clicked once, it displays the alternate title and button. After the button is clicked again, it returns to normal. This is the default for a radio or switch button.

Setting Button Titles

The Title field is the label that appears on most buttons. You can edit it two ways:

- In the Attributes display, edit the Title field
- In your interface, double-click the text on the button and type over it.



The alternate title (Alt. Title) is used only if the button type is Momentary Change or Toggle. If either of these types is selected, the alternate title replaces the title when you click the button, and the title replaces the alternate title when you click it again.In addition to the regular and alternate states, an NSButton object can actually have a third mixed state, but you cannot enable the third state from Interface Builder. You must add code to your application to do that.

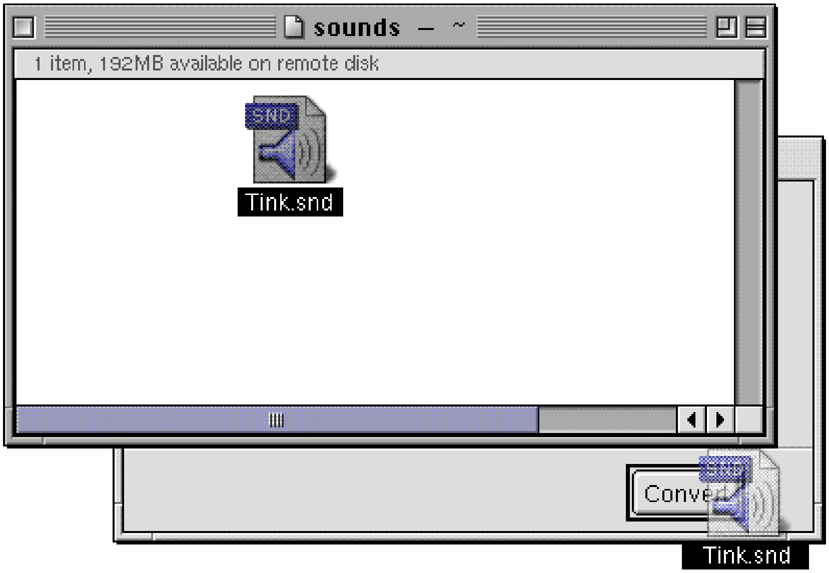
Setting Button Sound

If a sound is associated with a button, the application plays that sound whenever you click the button. There are two ways to associate a sound file with a button.

- If the sound file is already in the nib file, type its name in the Sound field.

  or

- Drag the file from the File Manager or the Sounds display of the nib file window directly onto the button. Interface Builder automatically enters the filename in the Sound field. If you dragged the file from the File Manager, Interface Builder also adds the file to the nib and it appears in the Sounds display of the nib file window.



Setting Button Icons

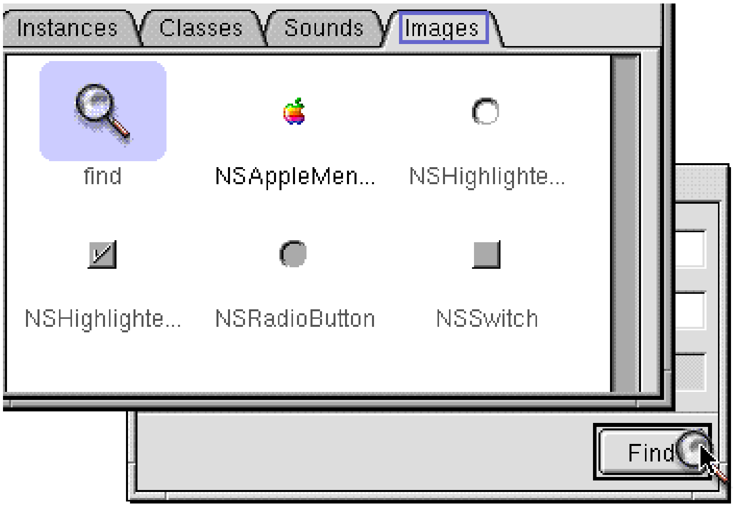
You can display an icon on your button with the Icon and Alt. Icon fields. The alternate icon (Alt. Icon) is used only if the button type is Momentary Change or Toggle. If either of these types is selected, the alternate icon replaces the icon when the user clicks the button, and the icon replaces the alternate icon when the user clicks it again.If you're using either a radio button or switch button (checkbox), the filename of the preferred icon for these button types is already filled in. Use the preferred icons if at all possible.

To display an icon on a button, enter the icon's filename in the Icon field. There are two ways to do this:

- If the icon file is already in the nib file, type its name into the Icon field.

  or

- Drag the file from the File Manager or the Images display of the nib file window directly onto the button. Interface Builder automatically enters the filename in the Icon field. If you dragged the file from the File Manager, Interface Builder also adds the file to the nib so it appears in the Images display of the nib file window



There's only one way to enter a filename in the Alt. Icon field: add the icon file to the nib file and type its name in the Alt. Icon field.In addition to the regular and alternate states, An NSButton object can actually have a third mixed state, but you cannot enable the third state from Interface Builder. You must add code to your application to do that.

These sections also describe how to change the appearance of a button:

"Setting the Pixel Inset" (page 15)

"Setting the Position of the Button Text" (page 16)

"Setting the Alignment of the Button Text" (page 17)

Setting the Pixel Inset

The Pixels Inset pop-up list lets you choose how many pixels separate a button from the button's nearest edge. You can choose from None, One, Two, or Three.

These sections also describe how to change the appearance of a button:

“Setting Button Icons” (page 14)

“Setting the Position of the Button Text” (page 16)

“Setting the Alignment of the Button Text” (page 17)

Setting the Position of the Button Text

The six buttons in the Icon Position group let you change the position of the button title and icon, as described below.

Icon	**Description**
☐ —	Icon is above the title. This is the default for a push button with an icon.
— ☐	Icon is below the title
☐	Icon appears alone, with no title.
☐–	Icon is to the left of the title. This is the default for a radio button or switch button (checkbox).
–☐	Icon is to the right of the title.
—	Title appears alone, with no icon. This is the default for a push button without an icon.

These sections also describe how to change the appearance of a button:

“Setting Button Icons” (page 14)

“Setting the Pixel Inset” (page 15)

"Setting the Alignment of the Button Text" (page 17)

Setting the Alignment of the Button Text

The three buttons in the Alignment group let you choose how text is aligned within a button.

Icon	Description
⇤	Text is aligned to the left side of the button. This is the default for a radio button or switch button (checkbox).
→|←	Text is centered in the button. This is the default for a push button.
⇥	Text is aligned to the right side of the button.

These sections also describe how to change the appearance of a button:

"Setting Button Icons" (page 14)

"Setting the Pixel Inset" (page 15)

"Setting the Position of the Button Text" (page 16)

Setting Keyboard Alternatives

This applies only to Yellow Box for Windows.

The Key field identifies a keyboard alternative to clicking the button. Common values are \e (Escape) for a Cancel button, \r (Return) for the default button, and any normal letter or number.A keyboard alternative is different from the mnemonic, which selects the button but doesn't click it.

Setting Button Options

The switch buttons in the Options group let you change the appearance and default behavior of a button.

Option	Description
Bordered	The button's border is outlined
Rounded	The button has rounded corners and a light gray background (more like Mac OS). If this is off, the button has square corners and a dark gray background (more like OpenStep). This option applies only to push buttons.
Transparent	The button does not draw itself and is invisible, but it does track the mouse and send its action.
Continuous	The button sends its action message continuously as long as the button is pressed. If this is off, the button sends its action message once when it's clicked.
Disabled	When the button is initialized, it's grayed out and cannot be clicked.
Selected	When the button is initialized, it's selected. This applies only to buttons in a matrix.

Making and Managing Connections

In Interface Builder, you establish that two objects communicate with each other by connecting them. These sections describes how to make and maintain action and outlet connections:

Connecting Objects

In Interface Builder, you establish that two objects communicate with each other by connecting them. There are two types of connections: outlets and actions. These sections describe how to make them:

Making Outlet Connections

Making Action Connections

Making Outlet Connections

1. In your interface or nib file window, select the object that will send messages.
2. Control-drag a connection to the object that will receive messages.
3. In the Connections display of the Inspector panel that appears, select an outlet, and

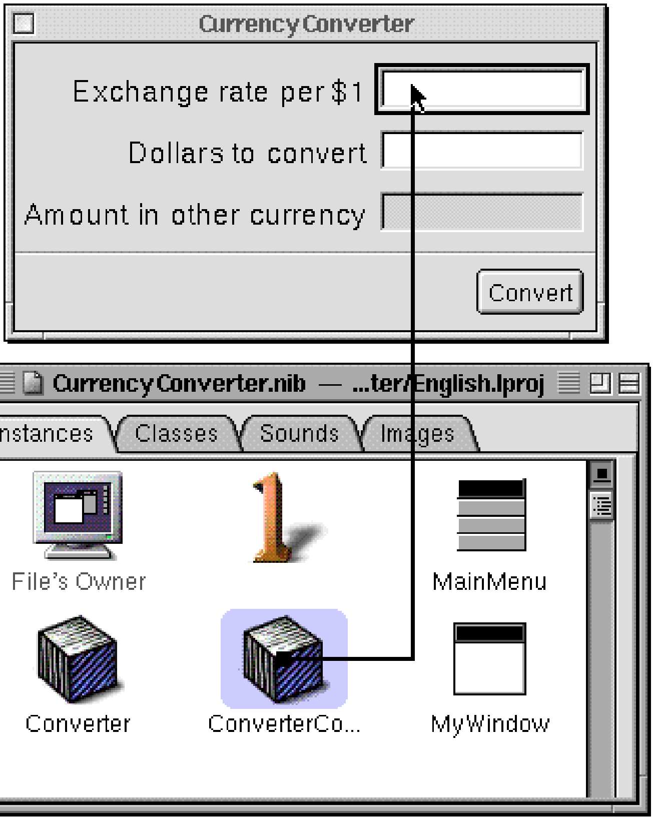
click Connect.

This section describes how to make outlet connections, so two objects can communicate with each other by sending messages. The object that sends messages contains a reference to the other object in an instance variable

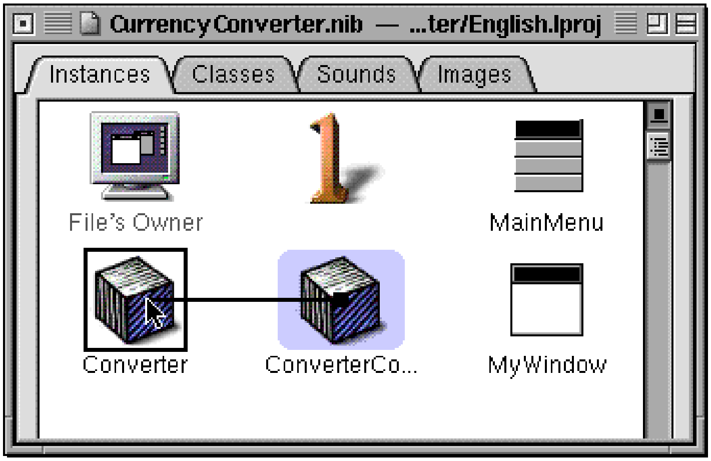
1. **In your interface or nib file window, select the object that will send messages.**

2. **Control-drag a connection to the object that will receive messages.**
   A reference to this object will be in an instance variable of the object selected above.

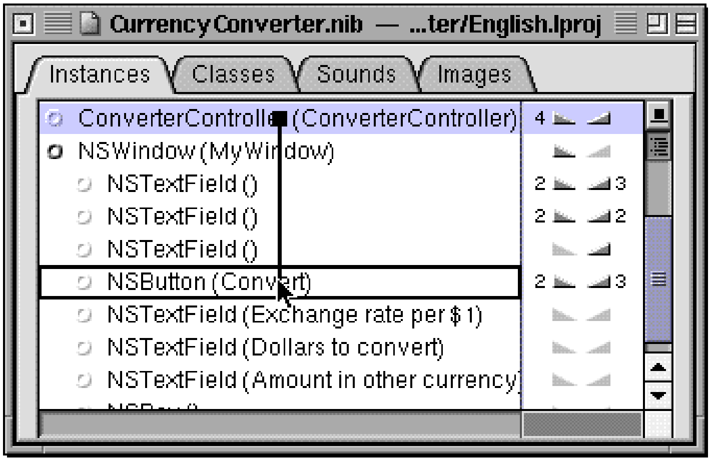
   Generally, you'll connect an object in the icon mode of the Instances display of the nib file window to an object on the interface.



You can also connect two objects in the Instances display of the nib file window.



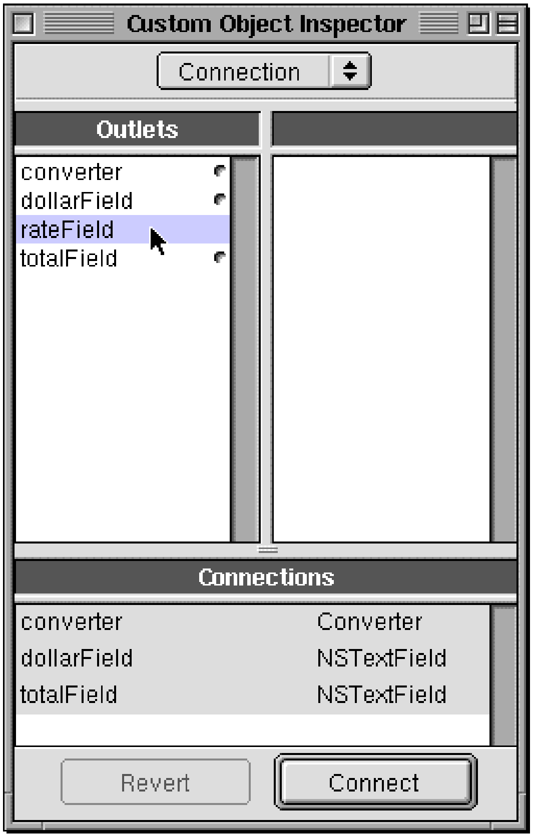
When you use the outline mode of the Instances display of the nib file window, you can connect any two objects within the nib file window, even if one of them is an object in your interface.


CurrencyConverter.nib — ...ter/English.lproj
Instances
Classes
Sounds
Images
ConverterController (ConverterController) 4
NSWindow (MyWindow)
NSTextField () 2 3
NSTextField () 2 2
NSTextField ()
NSButton (Convert) 2 3
NSTextField (Exchange rate per $1)
NSTextField (Dollars to convert)
NSTextField (Amount in other currency)

3. **In the Connections display of the Inspector panel that appears, select an outlet, and click Connect.**
Interface Builder displays the Connections display of the Inspector panel for the

source object (the object that the connection begins at). The first column lists the object's outlets. Already-connected outlets have a dimple to the right of them.

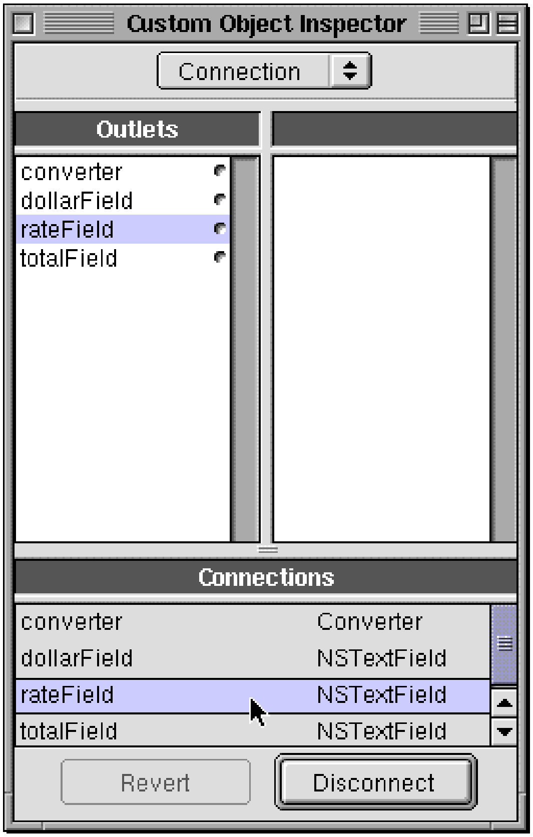
Select the appropriate outlet.



converter	Converter
dollarField	NSTextField
totalField	NSTextField

If the Connect button doesn't become active, you probably have connections locked. See "Locking connections" (page 35) for more information.

When you click the Connect button, the connection appears in the list at the bottom of the Connections display of the Inspector panel.



converter	Converter
dollarField	NSTextField
rateField	NSTextField
totalField	NSTextField

Make sure you connected the correct object and that you didn't, for example, connect to a form when you meant to connect to an item within the form.

See Also

"Nib File Window" (page 1)

"Outlet" (page 5)

"Locking connections" (page 35)

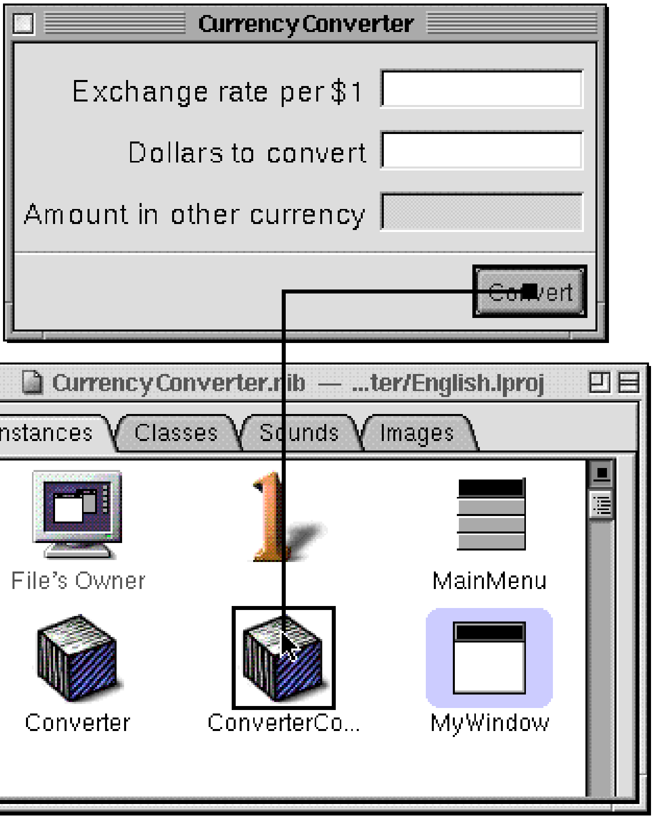
Making Action Connections

1. In your interface or nib file window, select the object that will send the action message.
2. Control-drag a connection to the object that will respond to the action messages.
3. In the Connections display of the Inspector panel that appears, select an action, and click Connect.

This section describes how to make an action connection, so an object in your interface can send action messages to another object. Action messages are sent whenever a user presses a button, selects a menu item, or manipulate an control object on your interface in any other way.

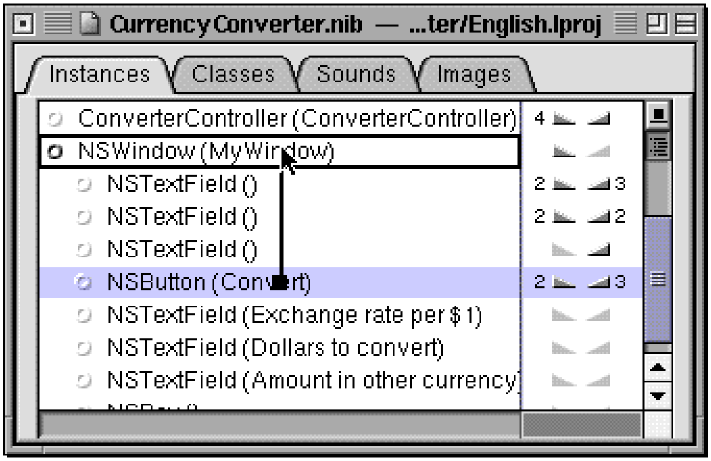
1. **In your interface or nib file window, select the object that will send the action message.**

2. **Control-drag a connection to the object that will respond to the action messages.**
   Generally, you'll connect an object in your interface to an object in the icon mode of the Instances display of the nib file window.



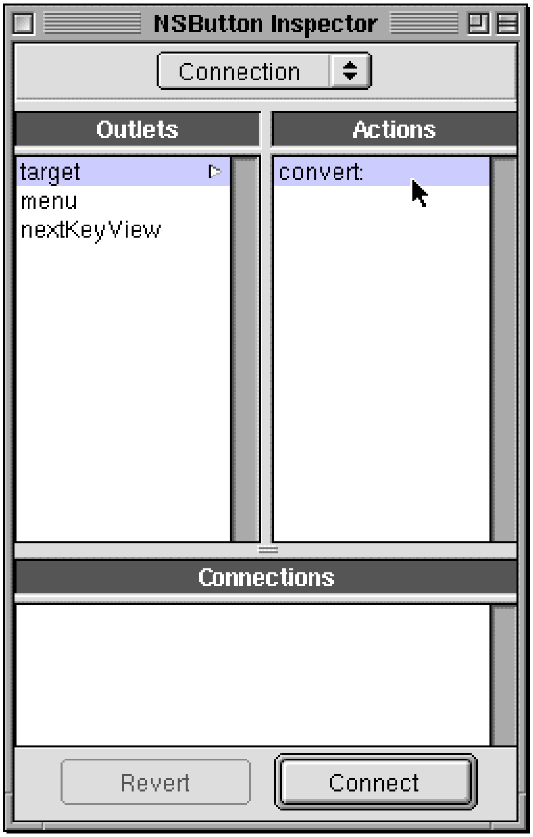
You can also connect two objects in the interface.

When you use the outline mode of the instances display of the nib file window, you can connect any two objects within the nib file window, even if they're in your interface.


CurrencyConverter.nib — ...ter/English.lproj
Instances
Classes
Sounds
Images
ConverterController (ConverterController)
NSWindow (MyWindow)
NSTextField ()
NSTextField ()
NSTextField ()
NSButton (Convert)
NSTextField (Exchange rate per $ 1)
NSTextField (Dollars to convert)
NSTextField (Amount in other currency)

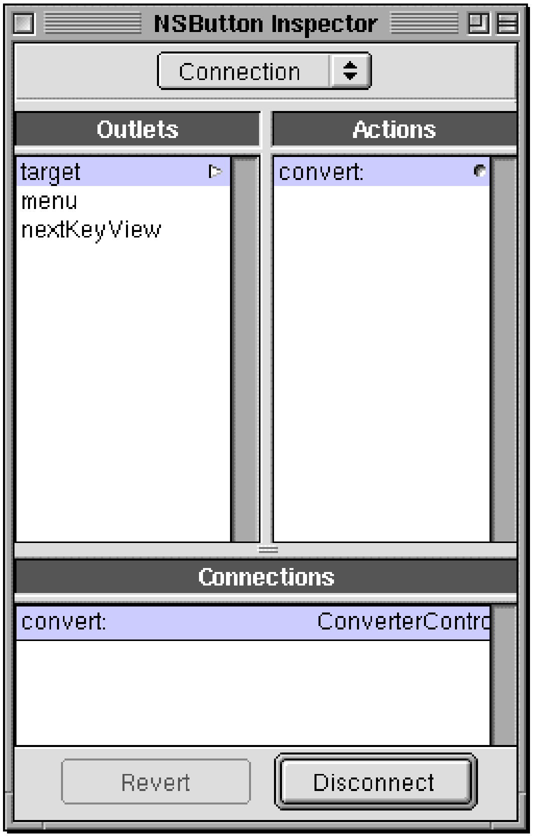
3. **In the Connections display of the Inspector panel that appears, select an action, and click Connect.**
   Interface Builder displays the Connections display of the Inspector panel for the source object (the object which the connection begins at). Make sure target is selected in the first column, so that the actions are listed in the second column.

   Select the appropriate action.



If the Connect button doesn't become active, you probably have connections locked. See "Locking connections" (page 35) for more information.

When you click Connect, the connection appears in the list at the bottom of the Connections display of the Inspector panel.



convert: ConverterContro

Make sure you connected the correct object and that you didn't, for example, connect to a form when you meant to connect to an item within the form.

See Also

"Nib File Window" (page 1)

"action" (page 6)

"Locking connections" (page 35)

Examining Connections

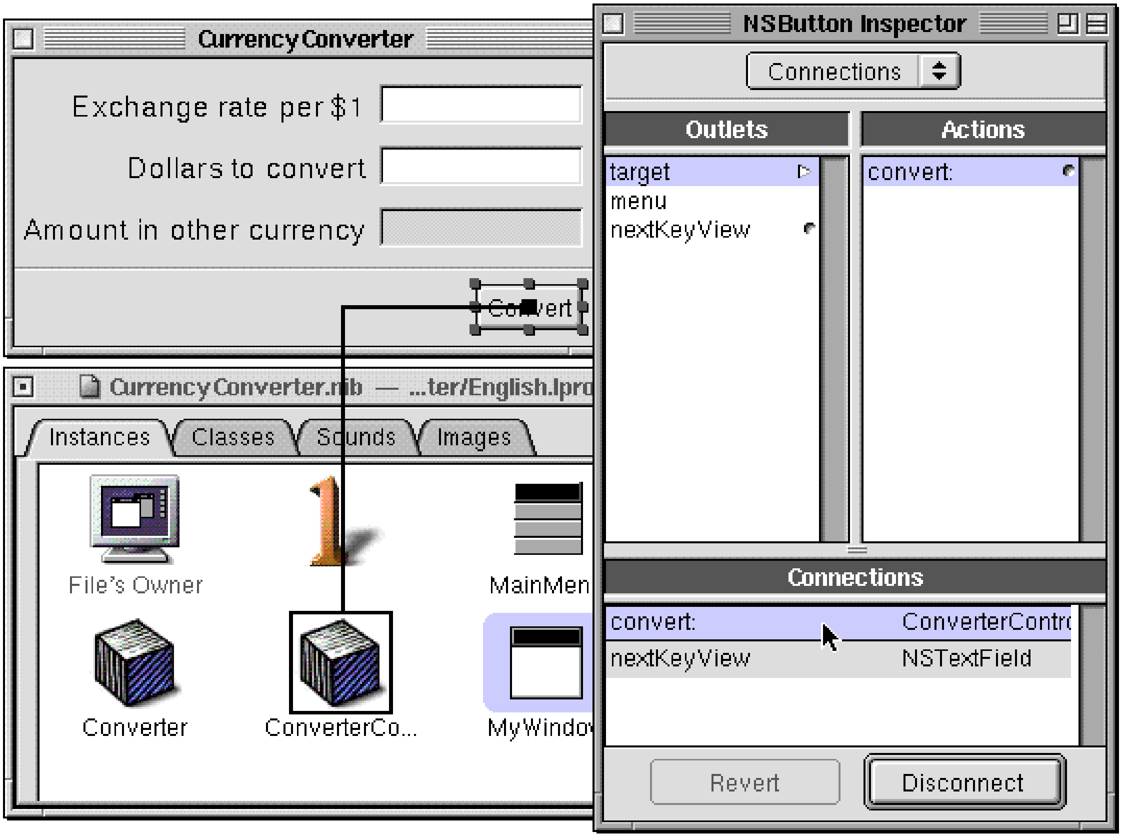
Interface Builder gives you a couple ways to examine connections between objects, so you can easily discover which outlets and actions are associated with an object in the interface. These sections describe how:

"Examining One Connection" (page 29)

"Examining All Connections" (page 31)

Examining One Connection

1. **In your interface or in the icon mode of the Instances display of the nib file window, select an object.**
2. **In the Connections display of the Inspector, select the outlet or action to view.**



convert:	ConverterContr
nextKeyView	NSTextField

1. **In your interface or in the icon mode of the Instances display of the nib file window, select an object.**

2. **In the Connections display of the Inspector, select the outlet or action to view.** To bring up the Connections display of the Inspector, choose Inspector from the Tools menu, and then choose Connections from the pop-up menu at the top of the Inspector. The bottom of the Inspector lists all the connections from this object.

   Select the outlet or action from the two lists at the top of the window. Interface Builder draws a line between the two objects.

See Also

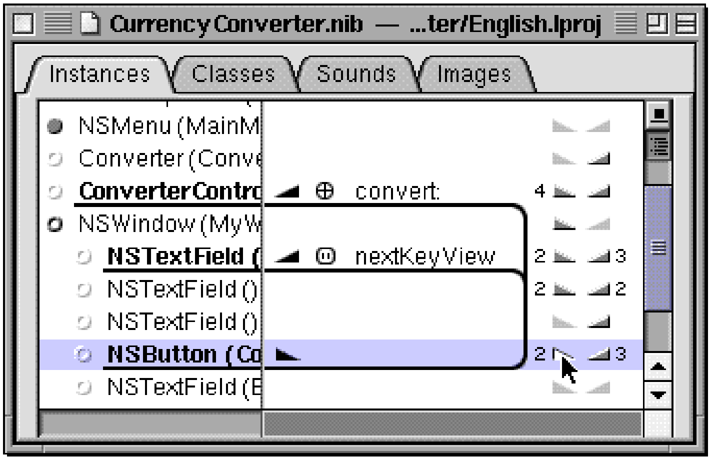
"Nib File Window" (page 1).

"Outlet" (page 5)

"action" (page 6)

Examining All Connections

1. In the outline mode of the Instances display of the nib file window, select the object.
2. Click one of the triangles to the right of the object.
3. If two objects have more than one connection between them, click the colon to view the other connections.

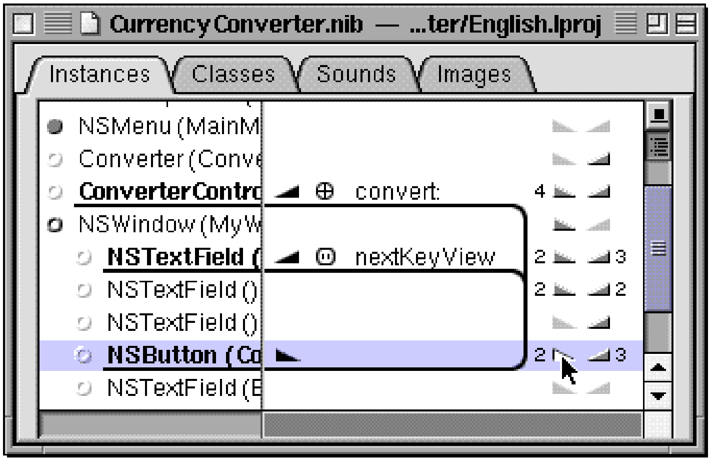


To examine all the connections to and from an object at once, follow this procedure.

1. **In the outline mode of the Instances display of the nib file window, select the object.**

2. **Click one of the triangles to the right of the object.**
   To see all the connections from this object, click the right-pointing triangle. To see all the connections into this object, click the left-pointing triangle.

   Lines appear on the nib file window, showing the connections between this object and other objects. The name and class of each connected object is highlighted in bold. Each connection is labeled with an icon showing whether this is an outlet or action and with the name of the action or outlet.



3. **If two objects have more than one connection between them, click the colon to view the other connections.**
   Sometimes, two objects are connected more than once; for example, they may share several outlets. In these cases, the connection's label lists which connection this is out of the total number of connections to this object; for example "1 : 3" means this is the first of three connections to this object. To cycle through the connections, click the colon between the two numbers.

   See Also

   "Nib File Window" (page 1).

   "Outlet" (page 5)

   "action" (page 6)

Disconnecting Objects

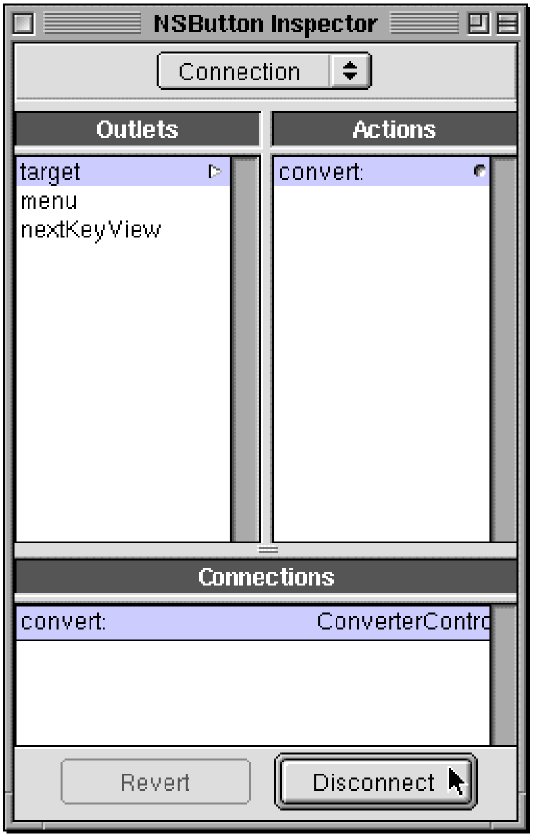
Interface Builder gives you a couple ways to break the connections between objects:

"Disconnecting Objects in the Inspector" (page 33)

"Disconnecting Objects in Outline Mode" (page 34)

Disconnecting Objects in the Inspector

1. **Select the object.**
2. **In the Connections display of the Inspector, select a connection and click Disconnect.**



This procedure describes how to disconnect two objects in the Inspector.

1. **Select the object.**
   Select the object, in either the nib file window or your interface.

2. **In the Connections display of the Inspector, select a connection and click Disconnect.**
   To display the Connections display of the Inspector, choose Inspector from the Tools menu, and choose Connections from the pop-up menu at the top of the Inspector that appears

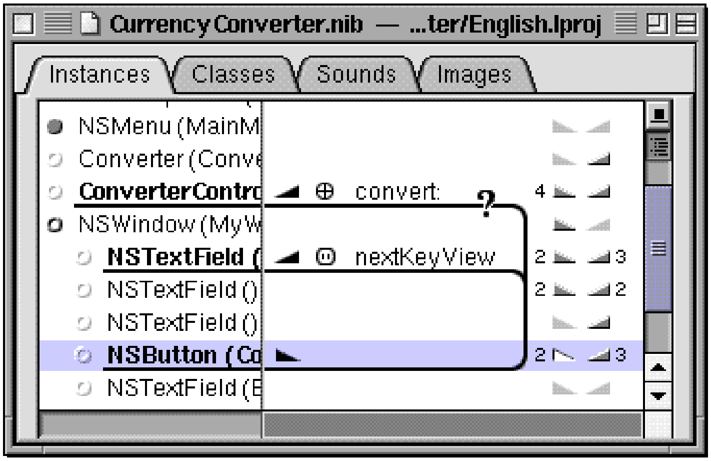
   All the object's connections appear in a list at the bottom of the Inspector. Select the one you want to delete, and click Disconnect. The connection disappears from the list.

   See Also

   "Nib File Window" (page 1)

Disconnecting Objects in Outline Mode

1. **In the outline mode of the Instances display of the nib file window, click one of the triangle buttons to the right of the object.**
2. **Control-click the connection line for the connection you want to delete**



This method describes how to disconnect two objects in the outline mode of the

Instances display.

1. **In the outline mode of the Instances display of the nib file window, click one of the triangle buttons to the right of the object.**
   To see all the connections from this object, click the right-pointing triangle. To see all the connections into this object, click the left-pointing triangle.

2. **Control-click the connection line for the connection you want to delete**
   As you hold down the Control key, the cursor should turn into a scissors. If it doesn't, make sure the cursor is in the column that contains the connections' names and icons. If you Control-click on the left side of the column, the click begins a connection operation.

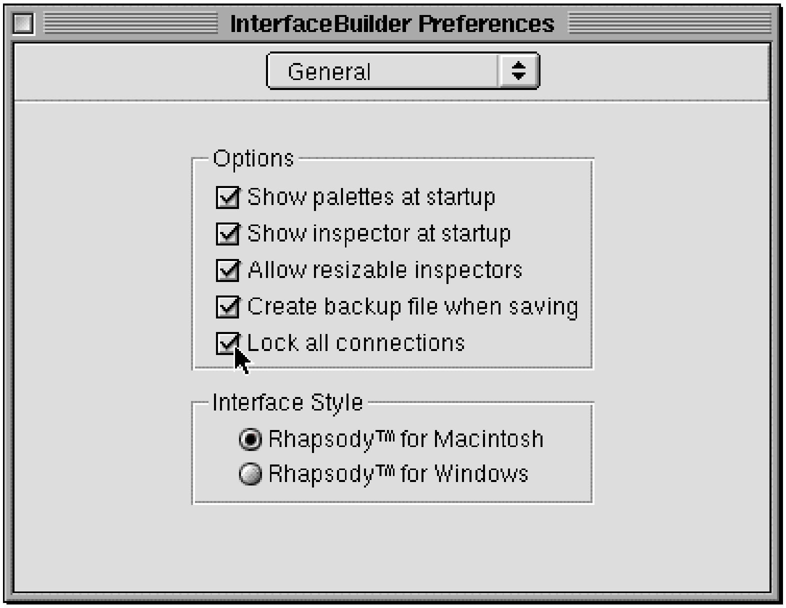
   After you Control-click, the connection is deleted and the line disappears.

   See Also

   "Instances Display of the Nib File Window" (page 1)

Locking connections

1. **In the General display of the Preferences panel, turn on the Lock All Connections option.**



Options

☑ Show palettes at startup

☑ Show inspector at startup

☑ Allow resizable inspectors

☑ Create backup file when saving

☑ Lock all connections

Interface Style

● Rhapsody™ for Macintosh

○ Rhapsody™ for Windows

Interface Builder lets you lock all the connections in your nib file, so you can’t accidentally delete them. For example, when you’re localizing an application, you want the interface objects to behave the same way, but you want their titles to change. Sometimes you need to move and resize the interface objects to make room for titles in other languages. You might accidentally delete an object, thereby deleting all its connections as well. By locking connections, you make sure that you can’t delete objects that have connections or make other changes that will affect how your application behaves.

1. **In the General display of the Preferences panel, turn on the Lock All Connections option.**
   To bring up the General display of the Preferences panel, choose Preferences from the Apple menu. The preferences panel appears. Choose General from the pop-up menu at the top of the panel.

   Click the Lock All Connections option so there’s a check in front of it.

Subclassing

Subclassing—Summary of Steps

You can use Interface Builder together with your code editor to make a new subclass. Within Interface Builder, you create the subclass, define how it works with the other objects in your application, and then generate template source files. Within your code editor, you write source code to flesh out the template.

For instructions on how to create a subclass, see:

"Creating a Subclass with Interface Builder—Summary of Steps" (page 37)

"Importing a Subclass into Interface Builder—Summary of Steps" (page 38)

Creating a Subclass with Interface Builder—Summary of Steps

The easiest way to create a subclass is to use Interface Builder.

You must do the tasks in the list below in a specific order. Almost every step requires something created in the step immediately before it.

1. **"Creating a New Class"**
2. **"Adding an Outlet"**
3. **"Adding an Action"**

4. **"Creating an Instance of Your Class"**
5. **"Connecting Your Class's Actions"**
6. **"Connecting your Class's Outlets"**
7. **"Making an Instance of Your Class a Delegate" (optional)**
8. **"Generating Source Code Files"**
9. **Implementing a subclass of NSObject (to be written)**
   or

   Implementing a subclass of NSView (to be written)

Importing a Subclass into Interface Builder—Summary of Steps

You can also import an already existing subclass into Interface Builder. This could be a subclass you created in a code editor or a subclass you previously created for another project.

You must do the steps in the list below in a specific order. Almost every step requires something created in the step immediately before it.

1. **"Creating a Class by Hand" (optional)**
2. **"Adding Existing Classes to Your Nib File"**
3. **"Creating an Instance of Your Class"**
4. **"Connecting Your Class's Actions"**
5. **"Connecting your Class's Outlets"**
6. **"Making an Instance of Your Class a Delegate" (optional)**

And if you modify a class's actions or outlets after importing it, to this:

1. **"Updating a Class Definition"**

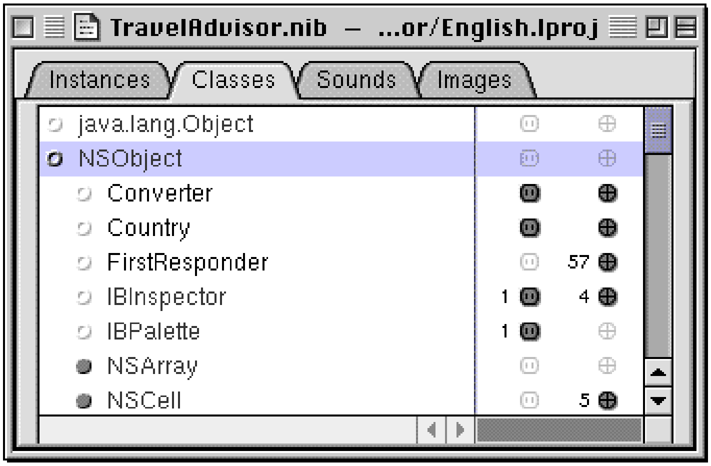
Subclassing—The Tasks

The following sections detail the individual steps for creating a subclass.

Creating a New Class

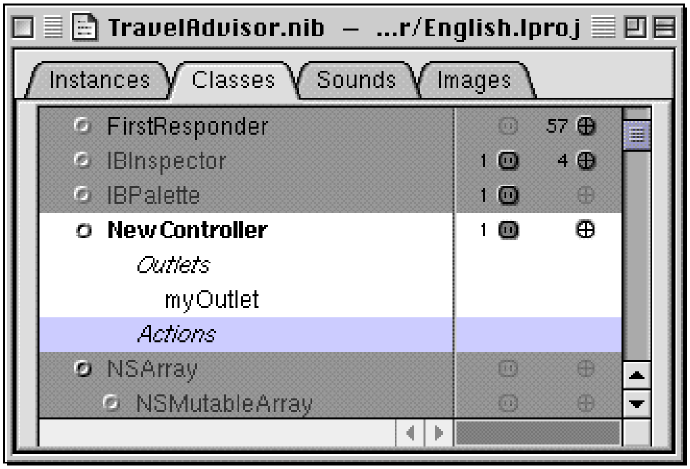
1. In the Classes display of the nib file window, select the class you want your class to inherit from.
2. Choose Classes>Subclass, or press Return.
3. Enter the name of your class.

1. In the Classes display of the nib file window, select the class you want your class to inherit from.



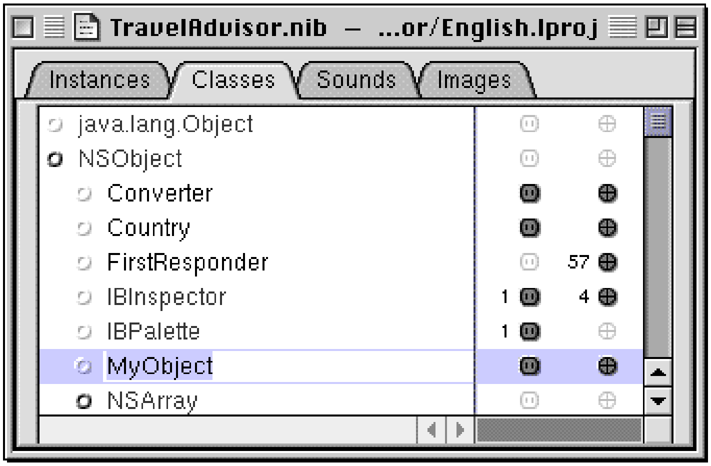
java.lang.Object		
NSObject		
Converter		
Country		
FirstResponder		57
IBInspector	1	4
IBPalette	1	
NSArray		
NSCell		5

2. Choose Classes>Subclass, or press Return.



The new class is listed under its superclass with a default name: the superclass name prefixed with “My” (for example, “My NSObject”).

3. Enter the name of your class.



Type the name of the class over the highlighted default name, and press Return.

You can rename the class later. When you want to rename a class, select the class name by double-clicking it, and then type the new name.

See Also

If you’re creating a new subclass, do this next	If your importing a subclass, do this next
“Adding an Outlet” (page 41)	Not applicable

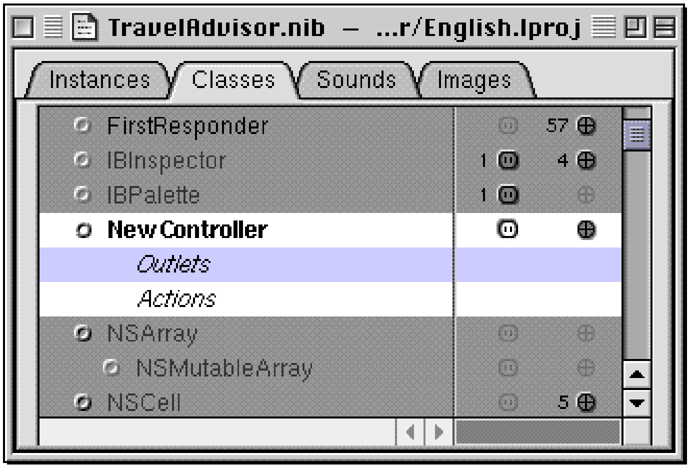
Adding an Outlet

1. In the Classes display of the nib file window, click the button for an outlet.
2. Choose Classes>Add Outlet.
3. Enter the name of your outlet.

An outlet is an instance variable that identifies another object.

1. **In the Classes display of the nib file window, click the button for an outlet.**
   In the Classes display of the nib file window, click the electrical-outlet button (⊡) beside your class’s name.

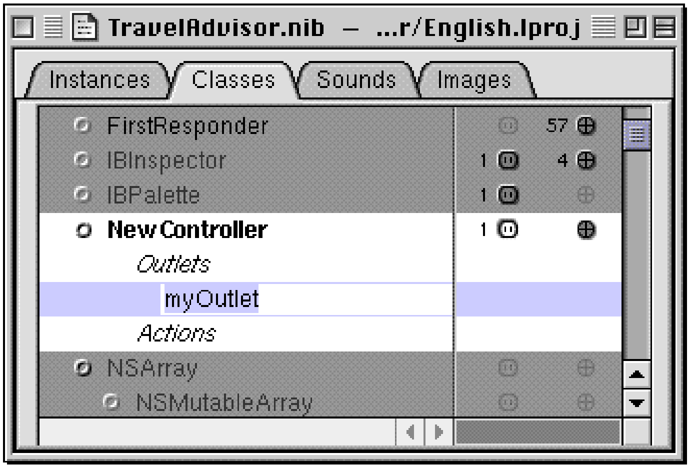
   The class’s actions and outlets appear underneath the class, with Outlets highlighted. If the class already has outlets, the first outlet is selected.



FirstResponder		57
IBInspector	1	4
IBPalette	1	
NewController		
Outlets		
Actions		
NSArray		
NSMutableArray		
NSCell		5

2. **Choose Classes>Add Outlet.**
   A new outlet appears, with the default name myOutlet selected.

3. **Enter the name of your outlet.**



FirstResponder		57
IBInspector	1	4
IBPalette	1	
NewController	1	
Outlets		
myOutlet		
Actions		
NSArray		
NSMutableArray		

Type the name of your outlet over the default name, and press Return.

To add more outlets, keep repeating steps 2 and 3. Each time you choose Add Outlet, it adds a new outlet to the end of the list.

See Also

"Nib File Window"

If you're creating a new subclass, do this next	If your importing a subclass, do this next
"Adding an Action" (page 42)	Not applicable

Adding an Action

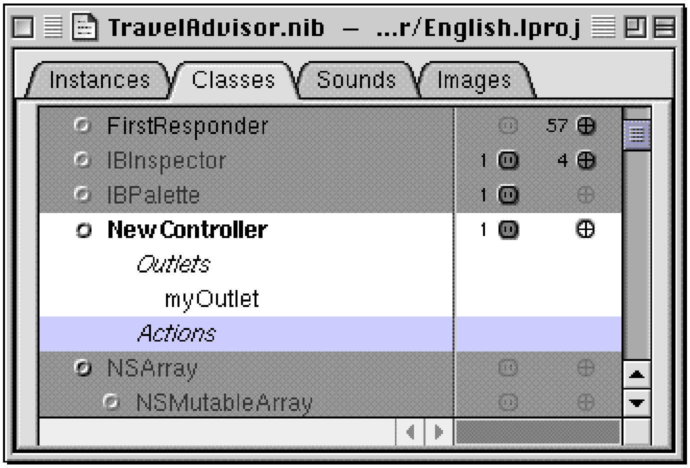
1. **In the Classes display of the nib file window, click the button for an action.**
2. **Choose Classes>Add Action.**

3. Enter the name of your action.

An action is a method that's invoked when a user manipulates an NSControl object in the interface, such as clicking a button.

1. **In the Classes display of the nib file window, click the button for an action.**
   In the Classes display of the nib file window, click the button for an action (⊕) beside your class's name

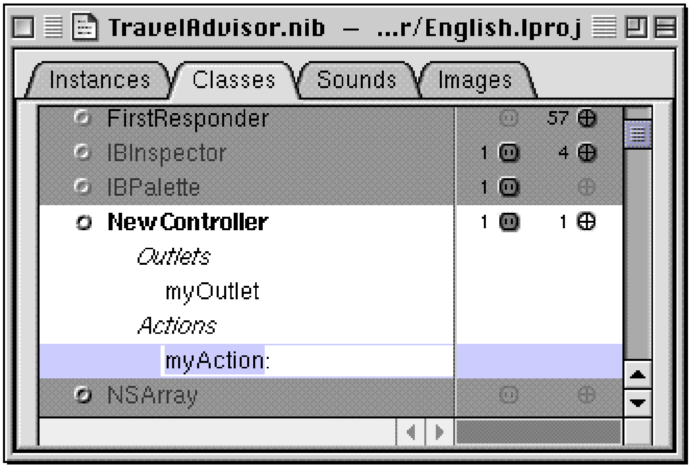
   The class's actions and outlets appear underneath the class, with Actions highlighted. If the class already has actions, the first action is selected.



2. **Choose Classes>Add Action.**
   A new action appears, with the default name myAction selected.

3. **Enter the name of your action.**
   Type the name of your action over the default name, and press Return.

   Interface Builder automatically adds a colon to the end of the name if you leave one out.



To add more actions, keep repeating steps 2 and 3. Each time you choose Add Action, it adds a new action to the end of the list.

See Also

"Nib File Window"

If you're creating a new subclass, do this next	If your importing a subclass, do this next
"Creating an Instance of Your Class" (page 44)	Not applicable

Creating an Instance of Your Class

Before you can specify how a new class interacts with the other classes in your application, you must create an instance of it.

Creating an Instance of a Non-NSView Class

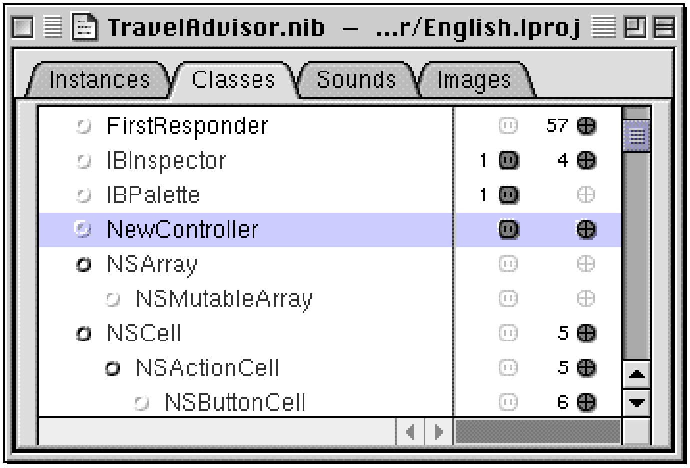
1. **Select your class in the Classes display of the nib file window, and choose**

Class>Instantiate..

This section describes how to create an instance of a class that doesn't have a user interface; that is, it doesn't inherit from NSView.

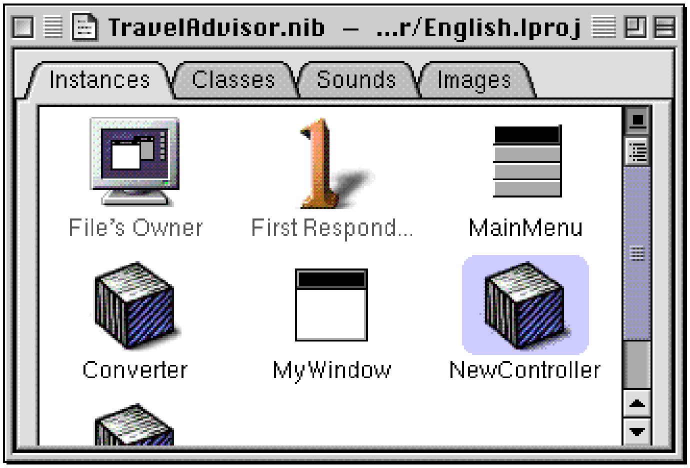
If you're creating a class that does inherit from NSView, see "Creating an Instance of an NSView Subclass" (page 46).

1. **Select your class in the Classes display of the nib file window, and choose Class>Instantiate.**



FirstResponder		57
IBInspector	1	4
IBPalette	1	
NewController		
NSArray		
NSMutableArray		
NSCell		5
NSActionCell		5
NSButtonCell		6

The nib file window switches to the "Instances Display of the Nib File Window". The new instance appears with the same name as the class.



See Also

"Nib File Window"

If you're creating a new subclass, do this next	If your importing a subclass, do this next
"Connecting Your Class's Actions" (page 49)	"Connecting Your Class's Actions" (page 49)

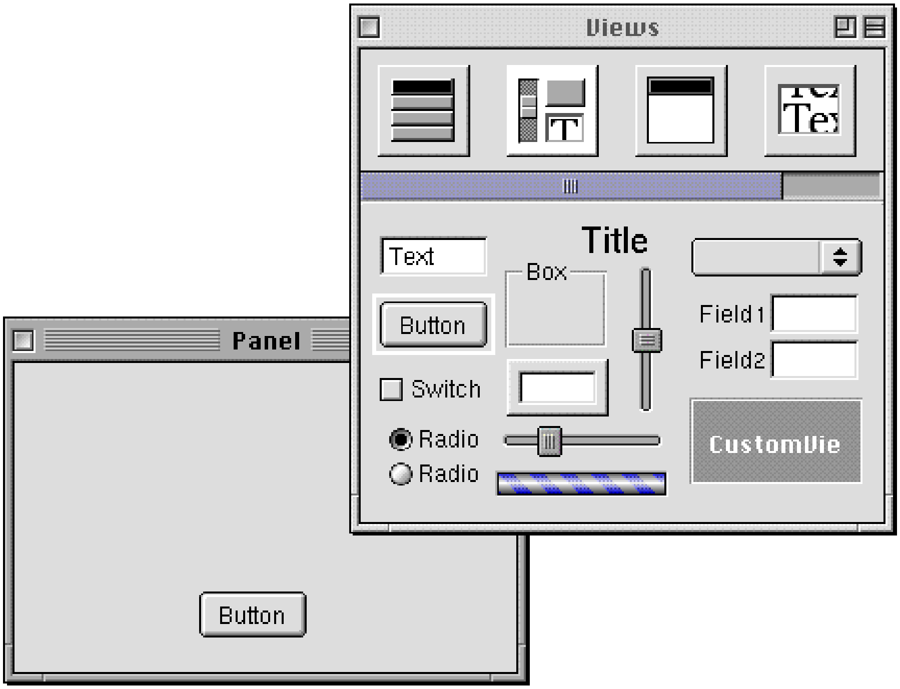
Creating an Instance of an NSView Subclass

1. From the Views palette, drag the object whose class you're subclassing onto your interface.
2. Position and resize the object as you want.
3. In the Custom display of the Inspector panel, choose your class as the object's class.

This section describes how to create an instance of a class that has a user interface; that is, it inherits from NSView.

If you're creating a class that does not inherit from NSView, see "Creating an Instance of a Non-NSView Class" (page 44).

1. **From the Views palette, drag the object whose class you're subclassing onto your interface.**
   If you're subclassing an NSView subclass (such as NSButton or NSTextField), drag the object that represents that class (that is, the button or the text field) from the palette window onto your interface's window. If you're subclassing NSView directly, use the CustomView object on the Views palette.

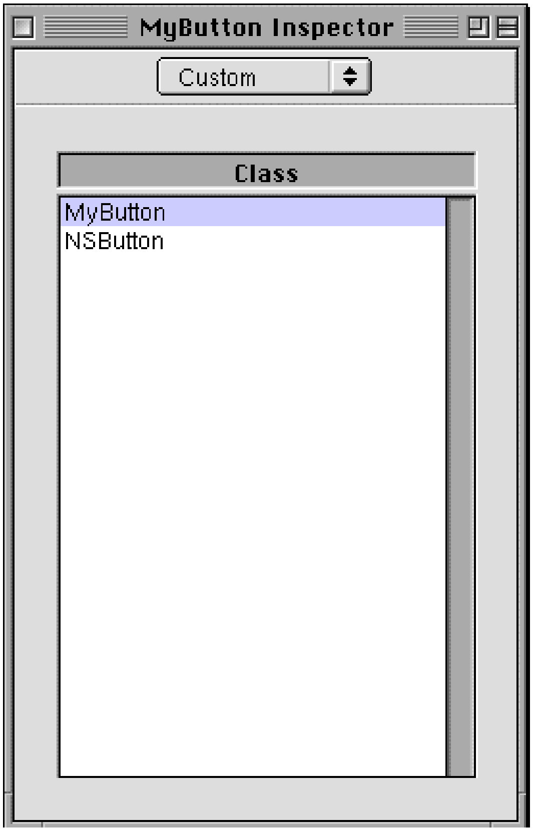


   Choose carefully which class you subclass. If you subclass NSView directly, you cannot set that class's attributes in the Inspector panel. If you subclass an NSView subclass, however, you can still set that class's attributes in the Inspector panel.

2. **Position and resize the object as you want.**

3. **In the Custom display of the Inspector panel, choose your class as the object's class.**

Choose your class name from the list of classes. The object is now a subclass of your new class.



See Also

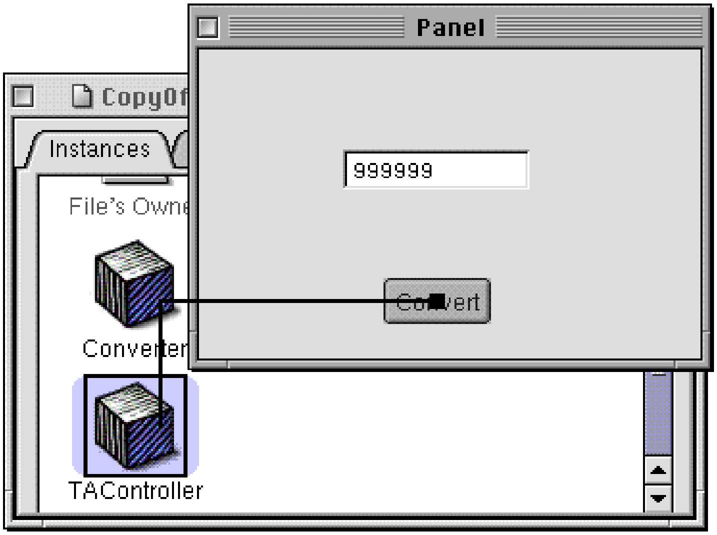
If you’re creating a new subclass, do this next	If your importing a subclass, do this next
“Connecting Your Class’s Actions” (page 49)	“Connecting Your Class’s Actions” (page 49)

Connecting Your Class’s Actions

1. Control-drag a line from a control object to another object.
2. In the Connection display of the Inspector panel, select the appropriate action, and click Connect..

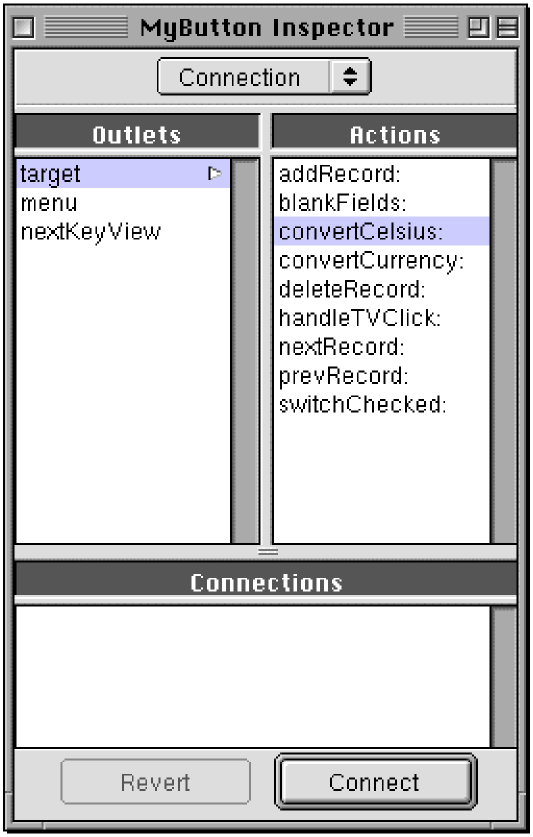
Project Builder lets you associate a control object in your user interface with an instance’s action. When your application runs, Yellow Box automatically invokes the action whenever the user clicks the Control.

1. **Control-drag a line from a control object to another object.**
   If your class does not inherit from NSView, the other object will be in the Instances display. If your class does inherit from NSView, the other object will be on your interface window.



Interface Builder brings the Connection display for the instance to the front. The second column of the Inspector panel lists the action methods declared for the instance.

2. **In the Connection display of the Inspector panel, select the appropriate action, and click Connect.**



Your new connection appears in the list at the bottom of the Inspector panel.

See Also

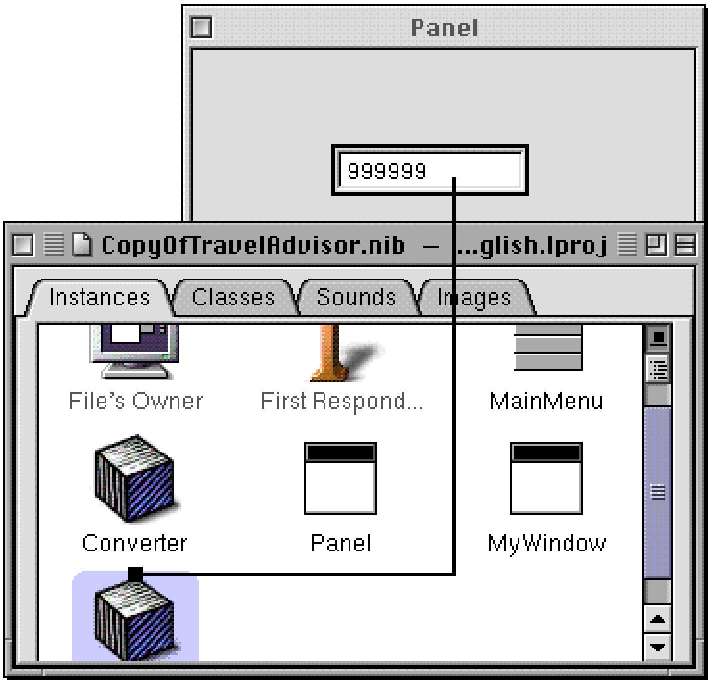
If you’re creating a new subclass, do this next	If your importing a subclass, do this next
“Connecting your Class’s Outlets” (page 52)	“Connecting your Class’s Outlets” (page 52)

Connecting your Class’s Outlets

1. **Control-drag a line from the instance to an item in your user interface.**
2. **In the Connections display of the Inspector panel, select the appropriate outlet, and click Connect.**

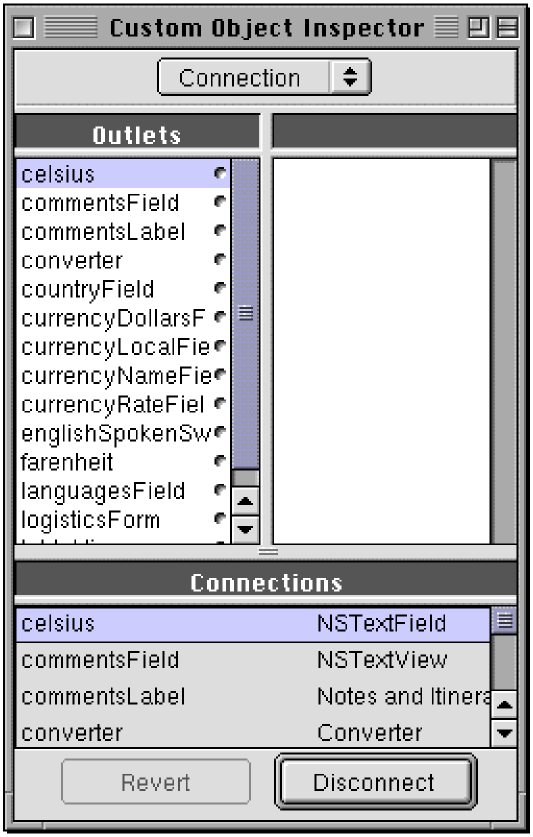
Project Builder lets you associate an object’s outlet with an item in your application’s user interface. When your application runs, Yellow Box automatically updates the outlet with whatever value the user enters in the user interface.

1. **Control-drag a line from the instance to an item in your user interface.**
   If your class does not inherit from NSView, the instance is in the Instances display. If your class does inherit from NSView, the instance is on your interface window.



Interface Builder brings the Connection display for the selected item to the front. The first column of the Inspector panel lists the outlets declared for the item.

2. **In the Connections display of the Inspector panel, select the appropriate outlet, and click Connect**
   Your new connection appears in the list at the bottom of the Inspector panel.



celsius	NSTextField
commentsField	NSTextView
commentsLabel	Notes and Itinera
converter	Converter

See Also

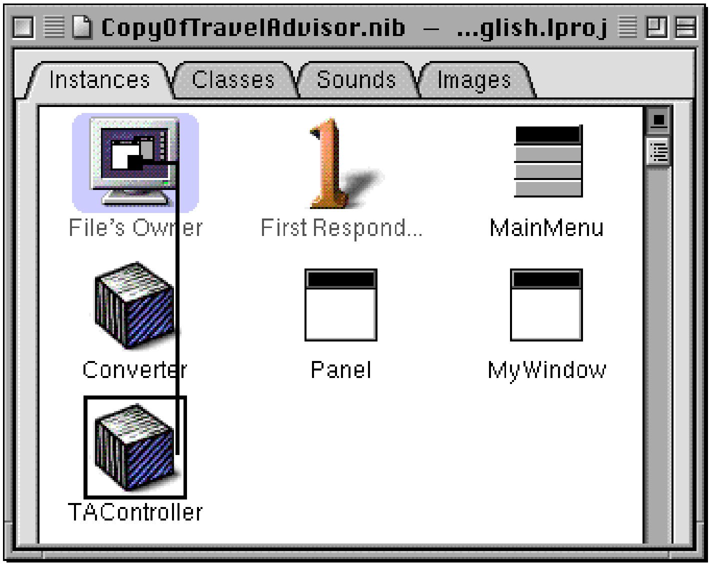
If you’re creating a new subclass, do this next	If your importing a subclass, do this next
“Making an Instance of Your Class a Delegate” (page 55)	“Making an Instance of Your Class a Delegate” (page 55)

Making an Instance of Your Class a Delegate

1. In the Instances display of the nib file window, connect your instance to an object that has delegates..
2. In the Connection display of the Inspector panel, select the delegate outlet, and click Connect.
3. Implement the delegate methods.

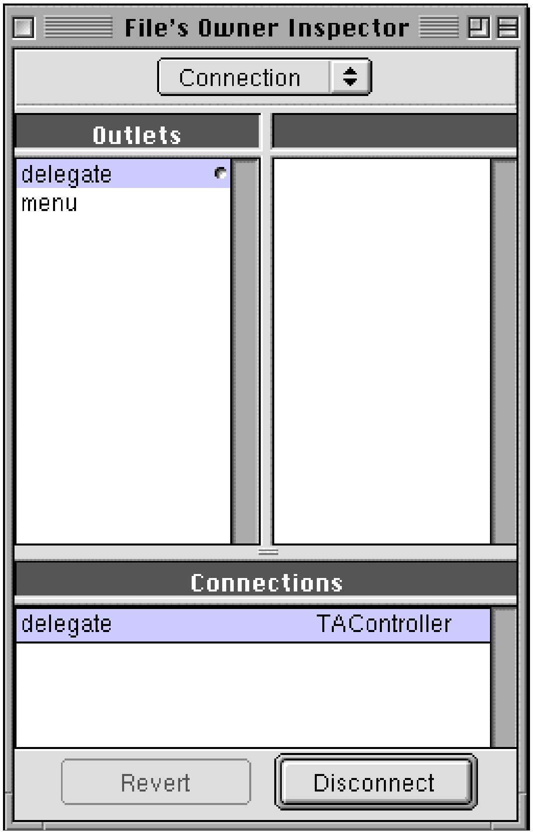
In Interface Builder, you can easily designate your class’s instance as a delegate.

1. **In the Instances display of the nib file window, connect your instance to an object that has delegates.**
   Control-drag from your object to the object with delegates.



The Connection display of the Inspector panel appears.

2. **In the Connection display of the Inspector panel, select the delegate outlet, and click Connect.**
   Your object is now a delegate of the other object.



delegate TAController

3. **Implement the delegate methods.**
   For more information on methods your object needs to implement, read the documentation for the class that defines the delegate method, such as NSApplication, NSWindow, NSText, or NSBrowser.

See Also

If you’re creating a new subclass, do this next	If your importing a subclass, do this next
“Generating Source Code Files” (page 58)	You’re done

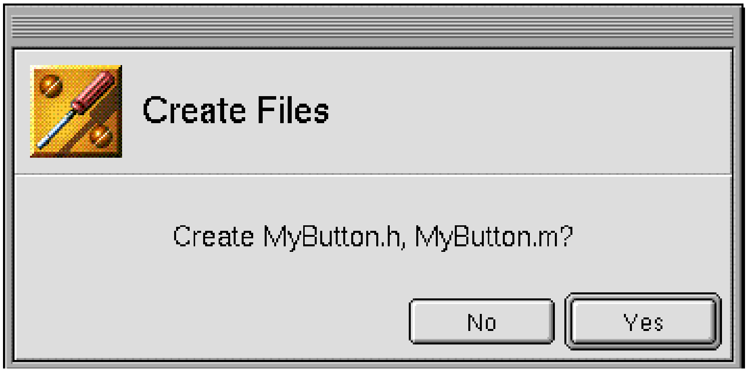
Generating Source Code Files

1. In the Classes display of the nib file window, select your class.
2. Choose Classes>Create Files, and click Yes in each of the two attention panels that appear.

Interface Builder can generate header files and implementation files for your class from the information in the nib file. The header file (MyClass.h) declares the outlets as instance variables (of type id) and declares the actions as instance methods (of the form *methodName*:`sender`). The implementation file (*MyClass*.**m**) contains empty function blocks for each of these methods.

1. **In the Classes display of the nib file window, select your class.**

2. **Choose Classes>Create Files, and click Yes in each of the two attention panels that appear.**
   First, Interface Builder displays an attention panel asking you to confirm the creation of the files. Click Yes.



Create Files

Create MyButton.h, MyButton.m?

Then, Interface Builder asks you whether to add the files to your project in Project Builder. Click Yes.

See Also

If you're creating a new subclass, do this next	If your importing a subclass, do this next
Implement the subclass (to be written)	Not applicable

Adding Existing Classes to Your Nib File

These sections describe how to add an existing class to the nib file you're working on:

"Adding an Existing Class by Dragging and Dropping" (page 59)

"Adding an Existing Class from a Nib File" (page 61)

"Adding an Existing Class from a Header File" (page 62)

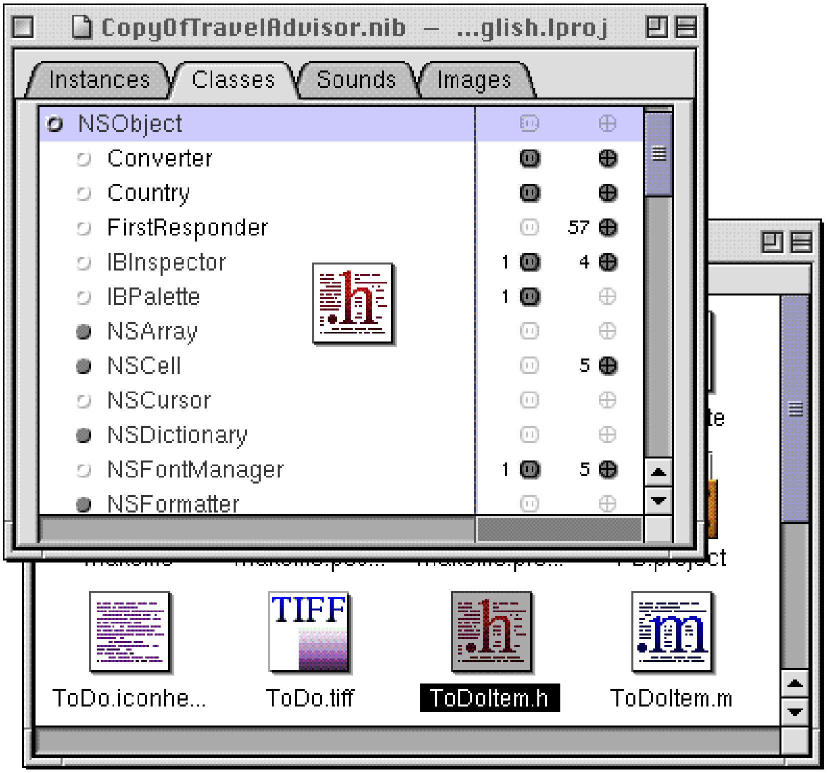
Adding an Existing Class by Dragging and Dropping

1. **Drag the header file for the class from the File Manager or Project Builder into the**

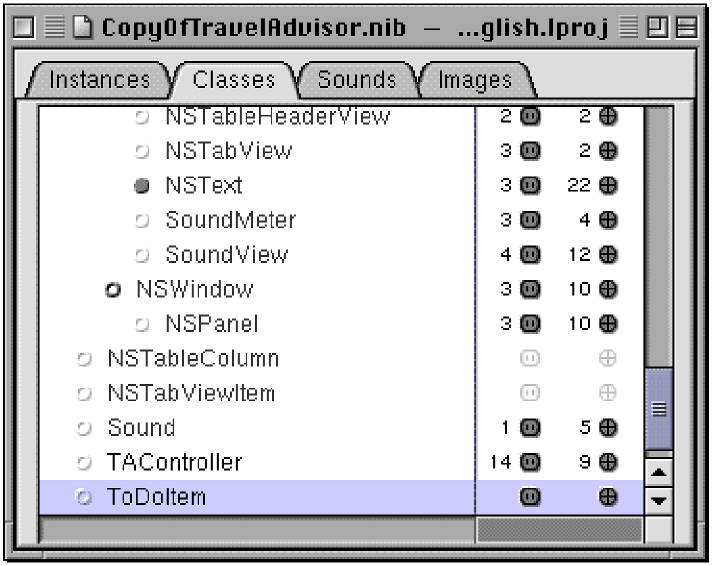
nib file window.

---

1. **Drag the header file for the class from the File Manager or Project Builder into the nib file window.**



The new class appears in the Classes display under its subclass, with its outlets and actions defined.



NSTableHeaderView	2	2
NSTabView	3	2
NSText	3	22
SoundMeter	3	4
SoundView	4	12
NSWindow	3	10
NSPanel	3	10
NSTableColumn		
NSTabViewItem		
Sound	1	5
TAController	14	9
ToDoItem		

See Also

If you’re creating a new subclass, do this next	If your importing a subclass, do this next
Not applicable	“Creating an Instance of Your Class” (page 44)

Adding an Existing Class from a Nib File

1. In the Classes display of one nib file window, copy the class you want to add to your current project.
2. In the Classes display of the destination nib file window, select the class’s superclass and paste the class.

1. In the Classes display of one nib file window, copy the class you want to add to your current project.

Select the class and choose Copy from the Edit menu.

2. **In the Classes display of the destination nib file window, select the class's superclass and paste the class.**
   Select the class's superclass and choose Paste from the Edit menu.

   A duplicate of the class appears in the Classes display under its subclass, with its outlets and actions defined.

See Also

If you're creating a new subclass, do this next	If your importing a subclass, do this next
Not applicable	"Creating an Instance of Your Class" (page 44)

Adding an Existing Class from a Header File

1. **Choose Classes>Read File.**
   Interface Builder displays an Open Panel.

2. **Choose the header file.**
   After you click OK, Interface Builder parses the header file and updates the nib file.

See Also

If you’re creating a new subclass, do this next	If your importing a subclass, do this next
Not applicable=	“Creating an Instance of Your Class” (page 44)

Creating a Class by Hand

1. Create the header file in a text editor, being sure to declare actions and outlets correctly.
2. Add the header file to your project.

You can create the source code for a class in any text editor, instead of defining a class in Interface Builder and using the Create Files command to create the source code.

1. **Create the header file in a text editor, being sure to declare actions and outlets correctly.**
   To create an action, declare a method that takes a single argument named sender and that returns void or that is coerced to be a action, like this:

   - (IBAction)foo: (id)Sender;

   To create an outlet, declare an instance variable that is of type Id or has the IBOutlet keyword prefixed to its declaration.

   If you create the header file in Project Builder, choose File>New in Project. Project Builder automatically creates a header file template for you. You need to add just the actions, outlets, and other member functions and variables.

   Here’s a sample header file:

```
#import <AppKit/AppKit.h>
@interface TAController:NSObject
{
    IBOutlet NSTableView *tableView;                  /* an outlet */
    id commentsLabel;                            /* another outlet */
    ...
```

```
        NSMutableDictionary *countryDict;            /* not an outlet */
        ...
    }
    /* target/action */
    - (void)addRecord:(NSButton *)sender;               /* an action */
    - (void)convertCelsius:(id)sender;            /* another action */
    /* Data read and write methods */
    - (void)populateFields:(Country *)aRec;        /* not an action */
    @end
```

2. **Add the header file to your project.**
   For more information, see "Adding Existing Classes to Your Nib File" (page 59)

See Also

If you're creating a new subclass, do this next	If your importing a subclass, do this next
Not applicable	"Adding Existing Classes to Your Nib File" (page 59)

Updating a Class Definition

1. **In the Classes display of the nib file window, select the class.**
2. **Choose Classes>Read File.**
3. **From the Open panel that appears, select the header file.**

If you edit the header file for a class outside of Interface Builder, you can update the nib file so it reflects any actions and outlets you added or deleted.

1. **In the Classes display of the nib file window, select the class.**

2. **Choose Classes>Read File.**
   Interface Builder displays an Open Panel.

3. **From the Open panel that appears, select the header file.**
   Interface Builder parses the header file and updates the nib file.