

ICS Tutorials: Menus and Dialogs

In this tutorial, you will add several pulldown menus and several types of dialogs to the address book. This will complete the principal elements of the interface.

Note: This tutorial assumes that you have chosen C as your language. The steps will be somewhat different if you are using ViewKit or Java. Those differences will be discussed in tutorials seven and eight respectively.

Before You Begin This Tutorial

- Make a new directory called `tutorial_4` and change directories to `tutorial_4`.
- If you have not started Builder Xcessory yet, enter the following at the prompt in the `tutorial_4` directory:

```
% bx
```

Start Builder Xcessory

Getting Started

Initialize your application as follows:

1. Clear the Builder Xcessory display by selecting **New** from the Main Window **File** menu.
2. Open the UIL file you saved at the end of Tutorial 3:

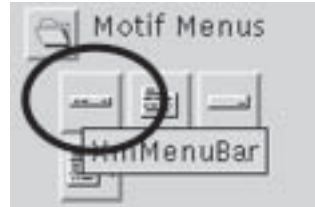
```
<tutorial_path>/tutorial_3.uil
```

Creating a Menu Bar

Before you create the pulldown menus, you must add a Menu Bar to the address book. To add the Menu Bar, perform the following steps:

1. Use MB2 to select `XmMenuBar` from the Motif Menus group of the Palette.
2. Drag the `XmMenuBar` to the Browser and drop it onto the `mainWindow` node in the tree. Builder Xcessory automatically places the menu bar across the top of the

`mainWindow`. The `XmMenuBar` will be empty because you have not added any menus to it yet! We'll do that next.

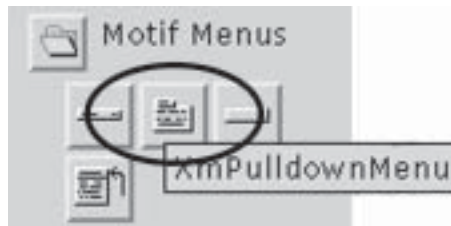


You will now add three pulldown menus to the address book: File, Edit, and Help.

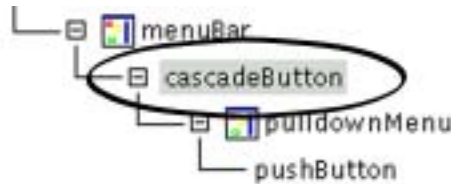
To create the File menu, perform the following steps:

Adding a PulldownMenu

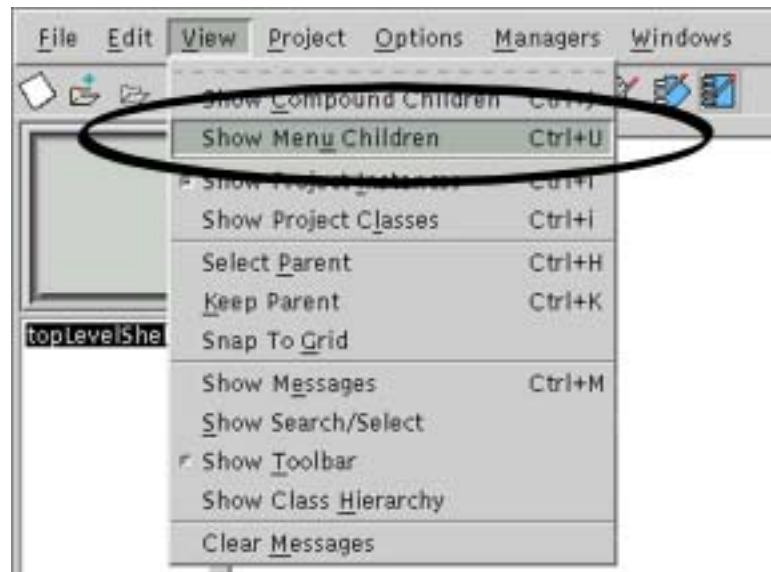
1. With MB2, select the `XmPulldownMenu` from the Palette and drop it onto the Menu Bar in the interface.



Note: When you create a `PulldownMenu`, Builder Xcessory actually creates 3 widget instances: a `CascadeButton`, with an associated `PulldownMenu`, which parents a `PushButton`. This hierarchy is reflected in the Browser.



2. If you have set the **Auto Menu Popup** toggle in the User Preference, the menu initially appears posted (i.e., “popped-up”). To unpost the menu, unset **Show Menu Children** in the **Main Window View** menu or press Ctrl/U, the keyboard accelerator for that menu item.



To change this default behavior, unset the **Auto Menu Popup** toggle in the **Behavior** tab of the **User Preferences** dialog. The **User Preferences** dialog is accessed via the **Options** menu choice in the Main Window.



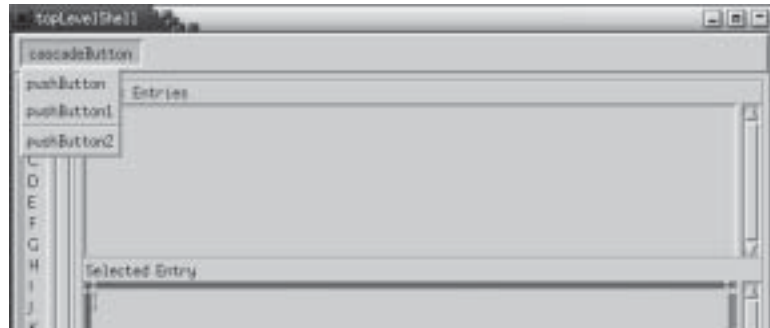
Adding Menu Items

Now you will add two more PushButtons and a Separator to the File menu. Perform the following steps:

1. Select `cascadeButton` by clicking MB1 on the node in the Browser.
2. If `pulldownMenu` is not already posted, post it by pressing Ctrl/U.
3. Use MB2 to drag an `XmPushButton` to the `pulldownMenu` on the interface. Builder Xcessory automatically adds items you place on a menu at the bottom of the menu.

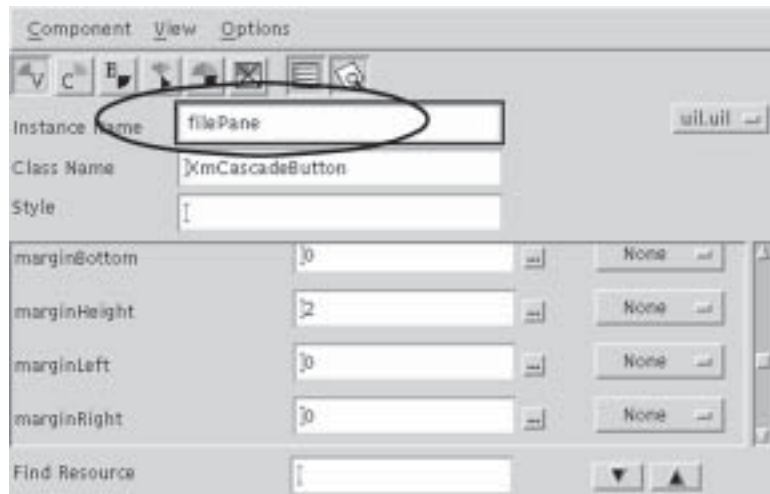


4. Add an `XmSeparator` to the menu.
5. Add another `XmPushButton` to the menu.



Naming Widget Instances

6. Select `cascadeButton` in the Browser.
7. Change the Instance Name to `filePane`.



8. Repeat steps 6–7 for each of the PushButtons, using the following names:

Object	New Name
pushButton	fileOpen
pushButton1	fileSave
pushButton2	fileExit

Labeling File Menu Components

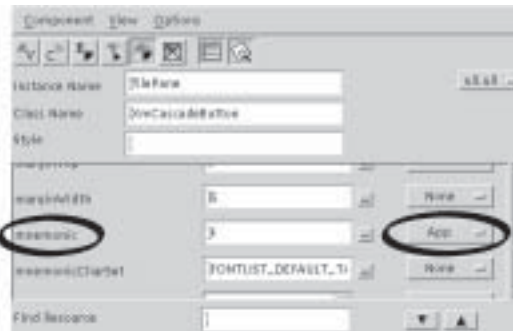
Now you will use the Resource Editor to add labels, accelerators, and mnemonics to the File menu. To label the File menu header, perform the following steps:

1. Select `filePane`.
2. Change the following resources, setting each storage location to **App**:

Resource	Value	Location
labelString	File	App
mnemonic	F	App

Note: Here and in the following tutorials, set the resource storage location for any displayed text to **App**. This will simplify internationalization in the future.

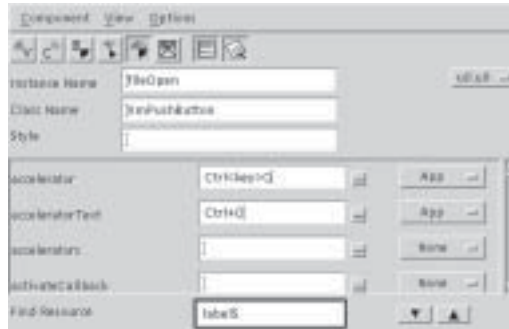
Note: Setting the mnemonic allows you to post the File menu by pressing Alt/F on the keyboard.



4. Similarly, set the following resources for `fileOpen`:

Resource	Value	Location
accelerator	Ctrl<key>O	App
acceleratorText	Ctrl+O	App
labelString	Open...	App
mnemonic	O	App

*Setting the
accelerator text
for fileOpen*



5. Set the following resources for `fileSave`:

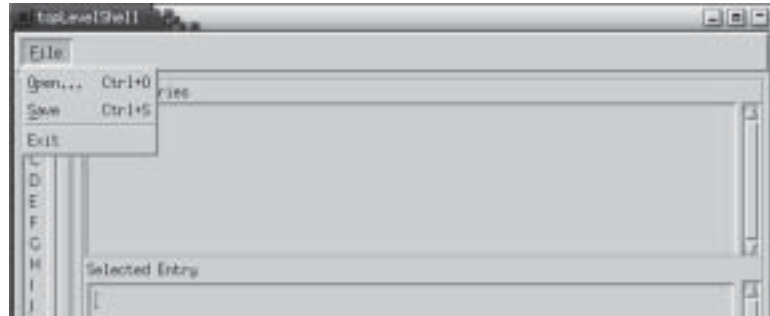
Resource	Value	Location
accelerator	Ctrl<key>S	App
acceleratorText	Ctrl+S	App
labelString	Save	App
mnemonic	S	App

6. Finally, set the following resources for `fileExit`:

Resource	Value	Location
labelString	Exit	App
mnemonic	X	App

Status Check!

Your interface should now appear like the figure below (remember to save your work in the `tutorial_4` directory before proceeding!)



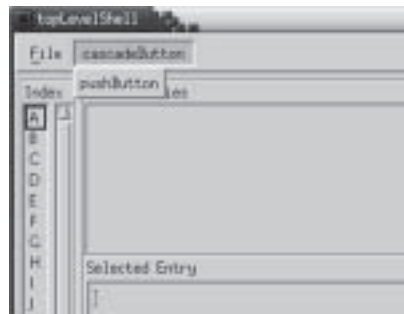
7. Select `filePane` again and hide the menu by pressing `Ctrl/U`

Using Keep Parent

Keep Parent makes it easier to add multiple children to a widget instance, especially when that widget instance is a menu.

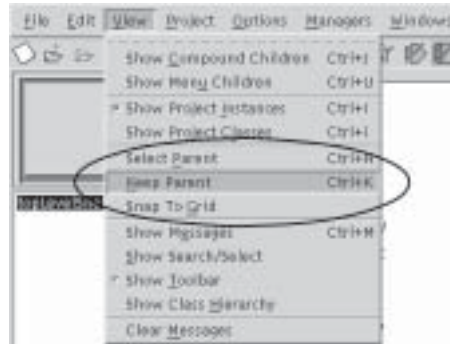
The Edit menu includes six `PushButtons` and a `Separator`. To add these elements, use the steps in the following sections:

1. Add an `XmPullDownMenu` to the interface by dragging it to the `menuBar` using MB2. The menu portion of your user interface should look like the image below.



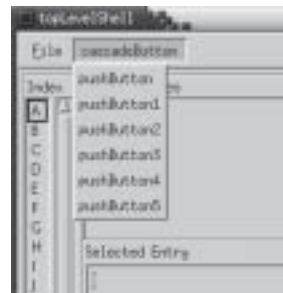
2. Select the **Keep Parent** toggle from the Main Window **View** menu.

Note: Only valid children for the selected object will be sensitive on the Palette.



Adding Menu Items

3. On the Browser, select `pullDownMenu1`.
4. Add five `XmPushButton`s. Because **Keep Parent** is on, you can add items to the interface simply by clicking on them in the Palette with MB1. Builder Xcessory knows to parent the new object to the selected object.

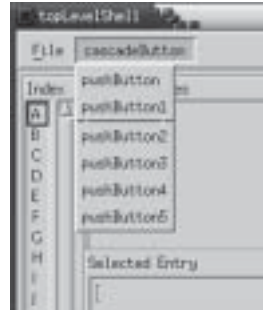


Moving a Menu Item

5. Add an XmSeparator to the interface. Click on XmSeparator on the Palette. The Separator is at the bottom of the menu. To move it up to the correct spot, perform the following steps:
6. Select separator1 on the Browser tree.
7. Press Ctrl/R to raise the Separator one level. (Ctrl/L lowers the item one level.)



8. Perform the raise operation three more times. separator1 should now follow pushButton1. Your menu should look like the following diagram.



9. Unset the **Keep Parent** toggle on the Main Window **View** menu.

Setting Instance Names for Edit Menu Components

As always, using your own instance names when working with Builder Xcessory is a good practice. Use the following table to assign new names for each of the objects in the Edit menu.

Object	New Name
<code>cascadeButton</code>	<code>editPane</code>
<code>pushButton</code>	<code>editNew</code>
<code>pushButton1</code>	<code>editModify</code>
<code>pushButton2</code>	<code>editCut</code>
<code>pushButton3</code>	<code>editCopy</code>
<code>pushButton4</code>	<code>editPaste</code>
<code>pushButton5</code>	<code>editDelete</code>

To change the Instance Name, perform the following steps for each object in the table above:

1. Click on the instance name in the Browser to select it. For example, click on `cascadeButton` in the Browser to select it.

2. Then in the Resource Editor window, click on the text `cascadeButton` to the right of the label “Instance Name”
3. Type the new name.
4. Click on the check mark at the end of the text field, or press Return.

Note: We set all the new names for the instances at once to avoid confusion. More commonly, you will assign a new Instance Name as you tailor the other resources of an object. We’ll use this more efficient approach in future tutorials.

Setting Resources for the Edit menu Components

Now you will use the Resource Editor to add labels, accelerators, and mnemonics to the Edit menu. Tailor the Edit menu by performing the following steps:

1. Select the `editPane` instance of the `cascadeButton` in the Browser.
2. Change the following attributes and resources:

Resource	Value	Location
labelString	Edit	App
mnemonic	E	App

3. Similarly, set the following attributes for `editNew` under `pulldownMenu1`:

Resource	Value	Location
accelerator	Ctrl<key>N	App
acceleratorText	Ctrl+N	App
labelString	New Entry...	App
mnemonic	N	App

4. Set the following attributes for `editModify` under `pulldownMenu1`:

Resource	Value	Location
accelerator	Ctrl<key>M	App
acceleratorText	Ctrl+M	App
labelString	Modify Entry...	App
mnemonic	M	App

5. Set the following attributes for editCut under pulldownMenu1:

Resource	Value	Location
accelerator	Ctrl<key>X	App
acceleratorText	Ctrl+X	App
labelString	Cut	App
mnemonic	t	App

Note: “C” is not available here, because mnemonics must be unique within a menu, and “C” is used for Copy. “t” is the value suggested by the *OSF/Motif Style Guide*.

6. Set the following attributes for editCopy under pulldownMenu1:

Resource	Value	Location
accelerator	Ctrl<key>C	App
acceleratorText	Ctrl+C	App
labelString	Copy	App
mnemonic	C	App

7. Set the following attributes for editPaste under pulldownMenu1:

Resource	Value	Location
accelerator	Ctrl<key>P	App
acceleratorText	Ctrl+P	App
labelString	Paste	App
mnemonic	P	App

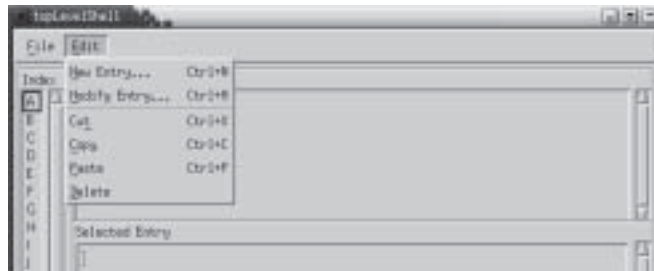
8. Set the following attributes for editDelete under pulldownMenu1:

Resource	Value	Location
labelString	Delete	App
mnemonic	D	App

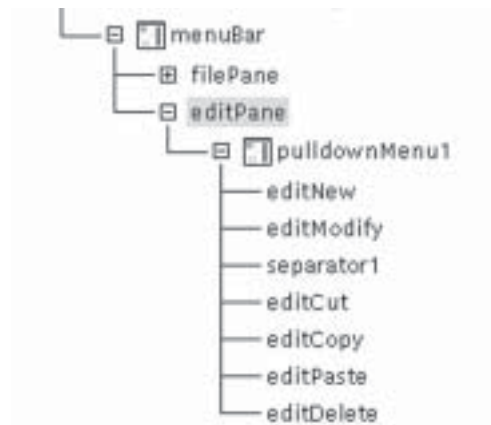
Note: There is no accelerator for the Delete function, since that would make a potentially damaging action too easy.

Status Check

The upper portion of your user interface should look like the sample below. You should compare your object instance hierarchy displayed in the Browser to the sample below. If they match, save your work and proceed!



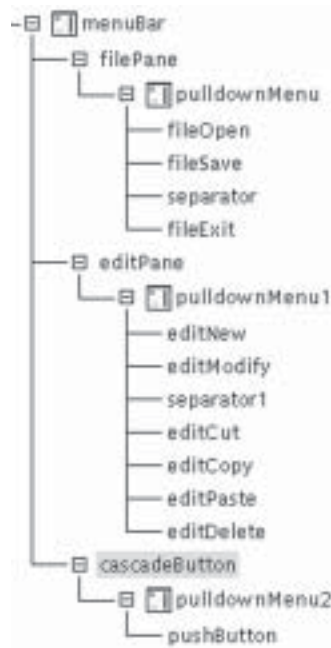
Structure of User Interface after setting resources for the Edit Menu



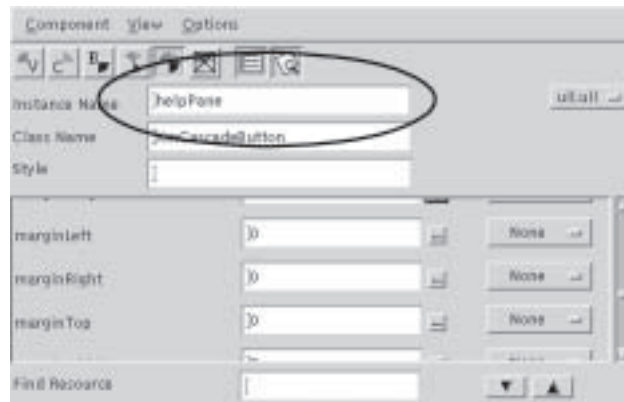
Adding the Help Menu

The Help menu includes only a single `PushButton`. To add these elements, use the following steps:

1. Add an `XmPulldownMenu` to the interface by dragging it to the `menuBar` using MB2. Three objects—`cascadeButton`, `pulldownMenu2`, and `pushButton`—appear on the Browser tree under `menuBar`.

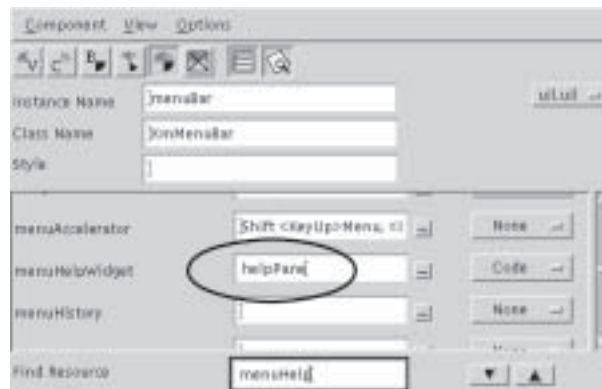


2. Select `cascadeButton` in the Browser (if not already selected).
3. In the Resource Editor window, click on the text `cascadeButton` to the right of the label “Instance Name”
4. Type the new name `helpPane`.
5. Click on the check mark at the end of the text field, or press Return.

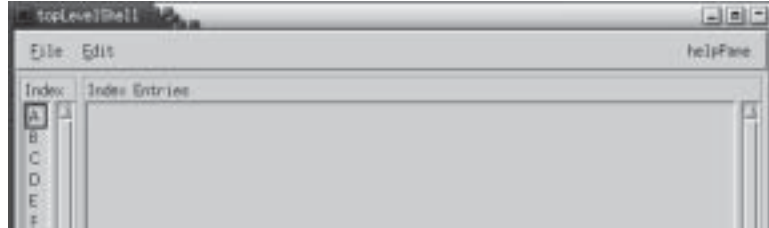


Setting Help Button

6. Select menuBar in the Browser.
7. Use the “Find” function of the Resource Editor to find the resource menuHelpWidget. Enter helpPane as the resource value.



Note: The help menu shifts to the far right edge of the Menu Bar.



Now you use the Resource Editor to add labels to the Help menu. To label the Help menu header, perform the following steps:

Labeling Help Menu Components

1. Select `helpPane` in the Browser.
2. Change the following resources:

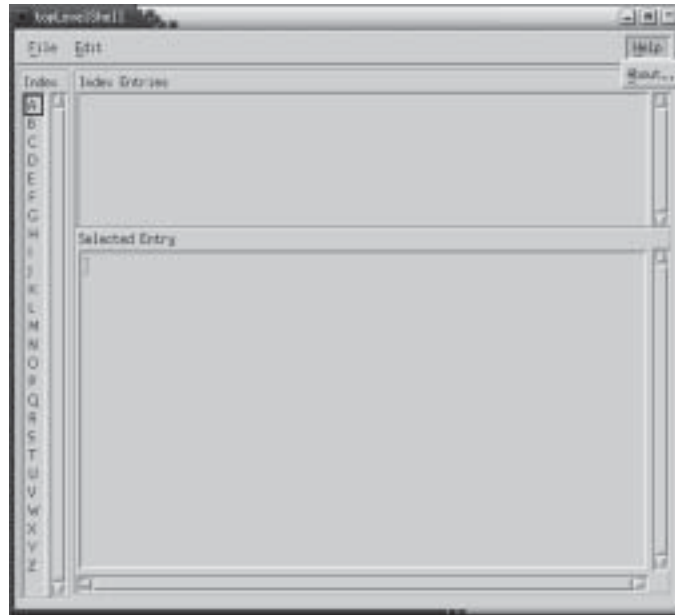
Resource	Value	Location
labelString	Help	App
mnemonic	H	App

3. Change the following resources for `pushButton`:

Resource	Value	Location
labelString	About	App
mnemonic	A	App

Status Check!

The basic interface is now complete and should appear like the figure below. If it does, now would be a good time to save your work.



Creating Dialogs

To complete the interface, you will now add three dialog boxes: for File Open, Edit New Entry, and Edit Modify Entry.

Adding a file selection box

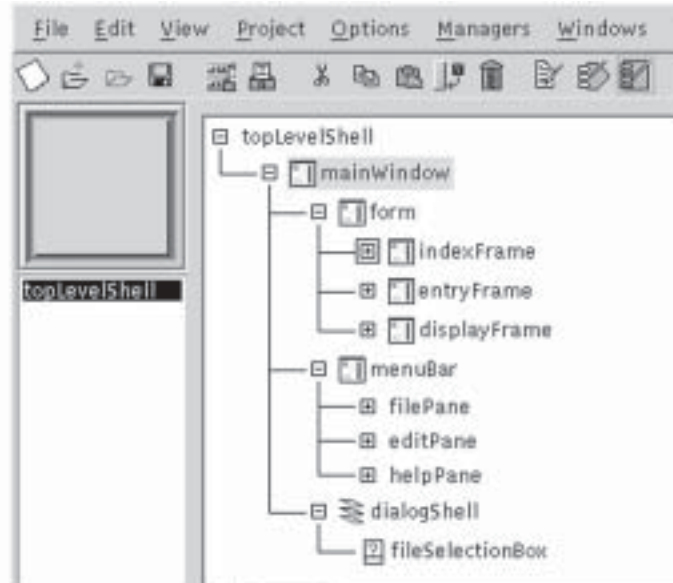
Motif provides a number of standard dialog widgets. For File Open, you will use a `FileSelectionBox`.

1. Select **Keep Parent** by clicking on the **View** menu in the Main Window.
2. Select `mainWindow` in the Browser with MB1.
3. On the Palette, click (and release!) on the `XmFileSelectionBox` with MB1. If it is grayed-out, then you do not have `mainWindow` selected.



4. Move the outline of the `XmFileSelectionBox` to your user interface.
5. Drop the `XmFileSelectionBox` anywhere on your user interface using Ctrl/MB3.

Note: Understand that you are using a two button mouse combination here. You click and release on the `XmFileSelectionBox` with MB1 and then move the outline over your user interface. Then you use the Ctrl/MB3 combination to drop it onto your user interface. The `XmFileSelectionBox` is named `fileSelectionBox` in the Browser tree. Using Ctrl/MB3 instead of MB1 to drop the object on the interface automatically sets it for “Natural Size,” and creates a `dialogShell` as well. You can only use this technique to drag an object to the actual interface, not to the Browser tree (as you can with MB2).



Make sure that the Widget Hierarchy looks like this and that the fileSelectionBox overlaps your user interface.

If it does not, then you may have parented the fileSelectionBox to the wrong place!



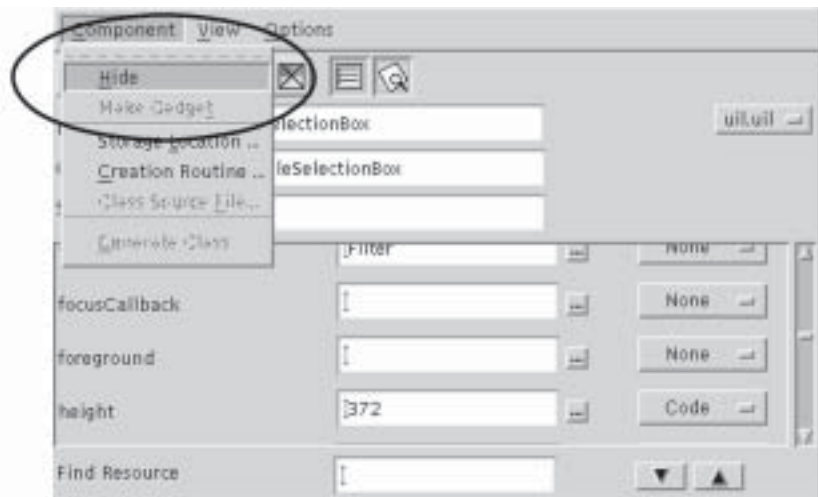
The `fileSelectionBox` becomes the child of the object where you dropped it. It overlaps the parent by default.

Warning: If you have not set **Keep Parent** and are not quite accurate in where you drop the `fileSelectionBox`, it might be created as an unparented `dialogShell`. Since it is not part of the main hierarchy tree, changes you make to the main tree, such as style changes, will not propagate automatically to the `dialogShell`.

Hiding an Object

To hide `fileSelectionBox`, perform the following:

1. Select the `fileSelectionBox`.
2. Select **Hide** on the Resource Editor **Component** menu.



Note: Hiding a user interface component performs two important functions. First it clears up screen clutter and allows you to focus on particular portions of the user interface. Second, when you generate code, “hidden” user interface components (typically dialog boxes) are created, but ***not managed***. This means that they will not be visible to the user of your application until your code specifically manages them.

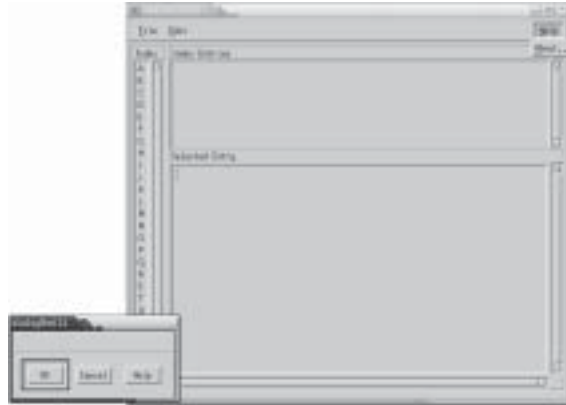
Adding a Message Box

To add an “About” dialog for the Help menu, perform the following steps:

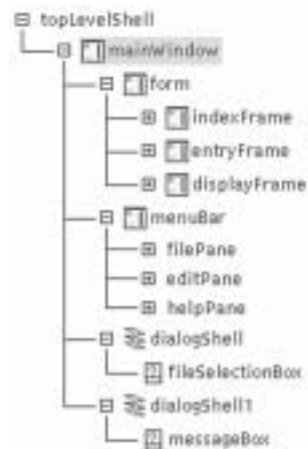
1. Confirm that **Keep Parent** is still selected as in the prior section.
2. Select `mainWindow` in the Browser with MB1.



3. On the Palette, click (and release) on `XmMessageBox` with MB1.
4. Move the outline of the `XmMessageBox` to the interface.
5. Drop the `XmMessageBox` anywhere on your user interface using the Ctrl/MB3 combination as in the previous section. The `XmMessageBox` is named `messageBox` in the Browser tree.

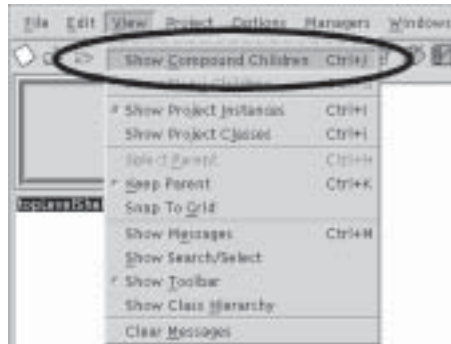


6. Compare your widget hierarchy with the following figure to confirm that the `XmMessageBox` was properly parented.

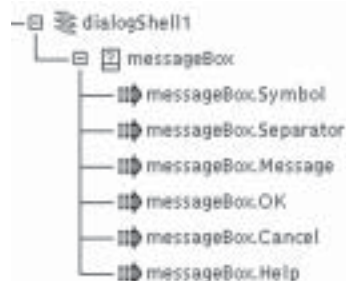


Viewing Compound Children

The MessageBox is a compound object, consisting of a number of smaller objects. To view the makeup of the Message Box, select **Show Compound Children** on the Main Window **View** menu. The Browser tree now displays the children of all compound objects in the interface.



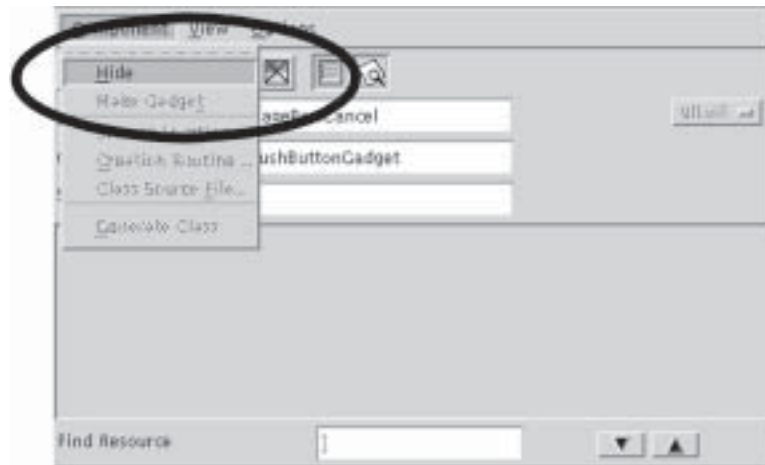
Note: Children of a compound object are indicated by a red arrow on the Browser tree. Builder Xcessory allows only limited modification of these children.



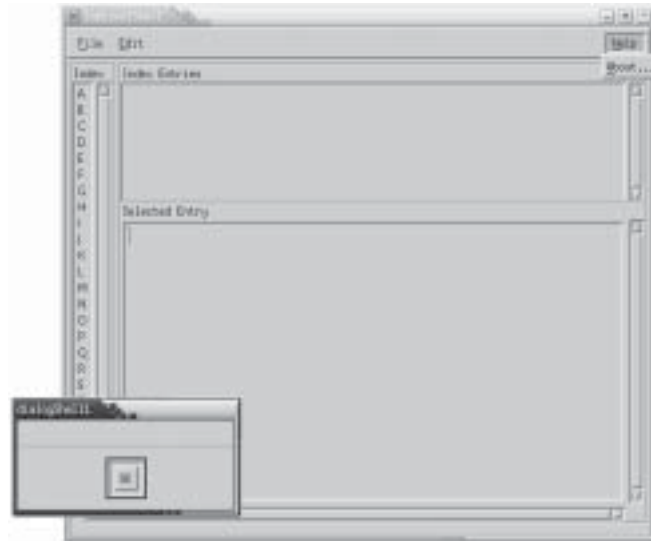
Removing Unused buttons

By default, the MessageBox contains three PushButtons labeled: OK, Cancel, and Help. For the “About” dialog, only the OK button is required, so you should delete the others as follows:

1. Select `messageBox.Cancel` on the Browser.
2. Select **Hide** on the Resource Editor **Component** menu. The Cancel button disappears from the display and will not be included when you generate code.



3. Repeat steps 1–2 for `messageBox.Help`.
4. To conceal the compound children, deselect **Show Compound Children** on the Main Window **View** menu.



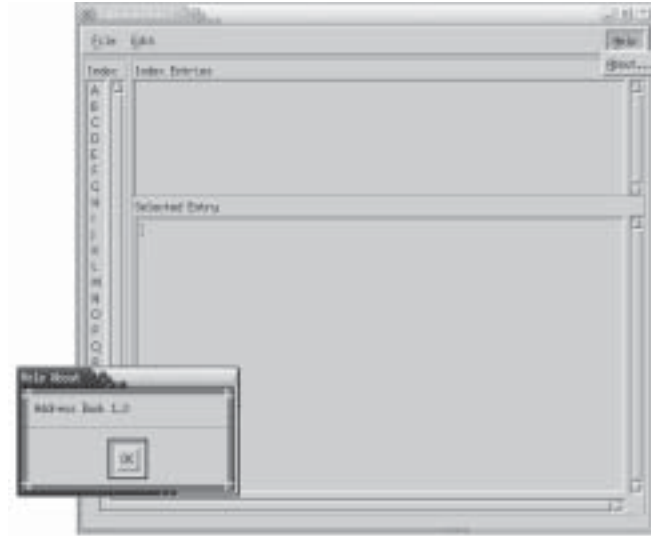
Labeling the About Box

To add text to the about box, set the following attributes:

1. Select `messageBox` in the Browser.
2. In the Resource Editor window, click on the text `messageBox` to the right of the label "Instance Name".
3. Type the new name `aboutBox`.
4. Set the remaining resources for `aboutBox` according to the table below. (Remember to set the resource placements to **App**!).

Resource	Value	Location
<code>dialogTitle</code>	Help About	App
<code>messageString</code>	Address Book 1.0	App
<code>okLabelString</code>	OK	App

5. Select `AboutBox` in the Browser



The aboutBox should be like the figure above

6. Select **Hide** on the Resource Editor Component menu

Customizing a Dialog

You can customize the standard dialog boxes to create dialogs specific to your application.

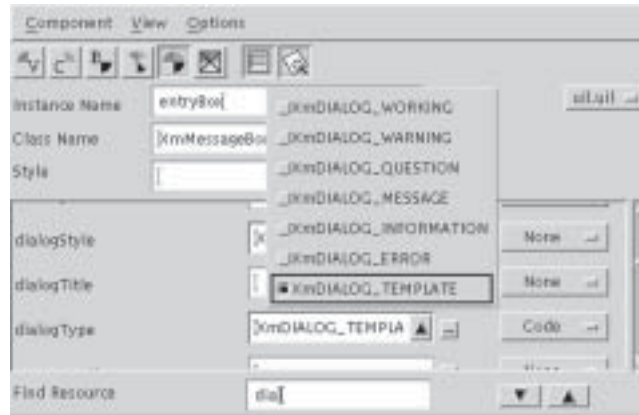
To create a customized dialog box, perform the following steps:

1. Click on `XmMessageBox` in the Palette with MB1.
2. Use `Ctrl/MB3` to place the `XmMessageBox` in the interface. (Remember to make sure **Keep Parent** is selected and that you have `mainWindow` selected in the Browser!) This `MessageBox` is named `messageBox` on the Browser tree.

Note: Since this is the second `MessageBox` you have added, it would normally be named `messageBox1`. However, you have already renamed the first `MessageBox` to `aboutBox`.

3. Select `messageBox` if it is not already selected.
4. Set the Instance Name to `entryBox` in the Resource Editor.

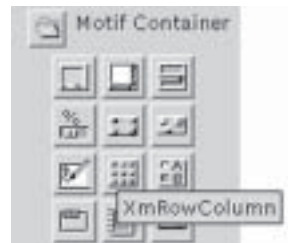
5. Set the `dialogType` to `XmDIALOG_TEMPLATE`.
6. Set storage location to **Code** (not **App!**).



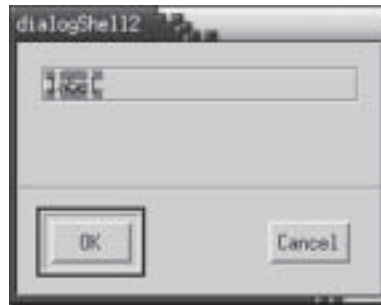
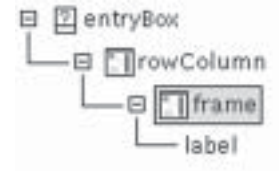
By default, the `XmMessageBox` contains three `PushButton`s labeled: OK, Cancel, and Help. When you reset the `dialogType` resource, those buttons disappear, and the dialog is blank.

You will now create a dialog box for making an address book entry. To add the necessary text entry fields, perform the following steps:

1. Select `entryBOX` (keep focus)
2. Place an `XmRowColumn` in `entryBOX`.



3. Place an `XmFrame` in `rowColumn`.
4. Add an `XmLabel` to `frame`.



Note: Because `frame` is so small in the interface, it is easier and more reliable to add the `XmLabel` on the Browser, or by setting **Keep Parent** on `frame`.

4. Set the following resources for `label`:

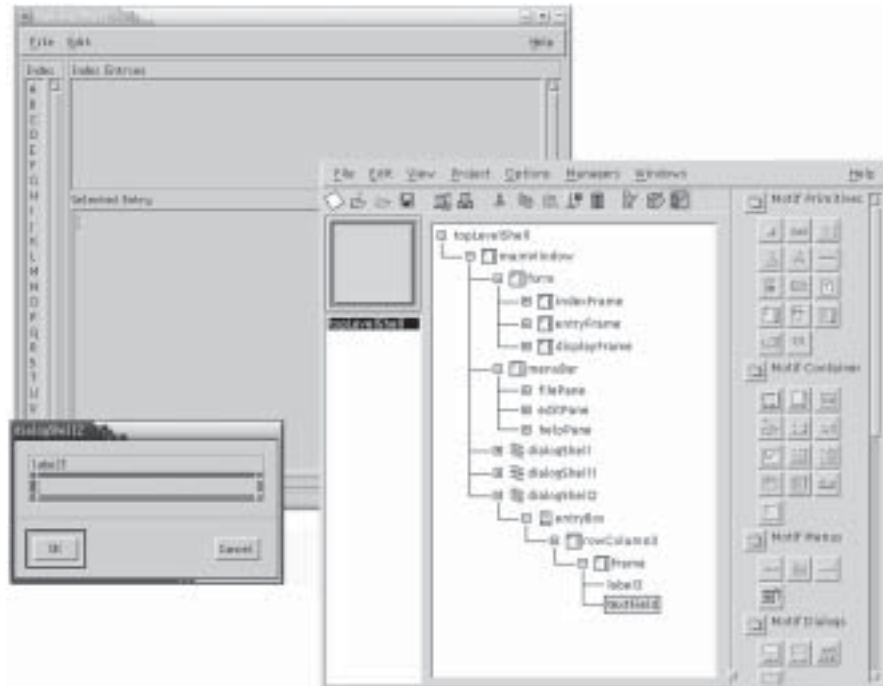
Resource	Value	Location
<code>childHorizontalSpacing</code>	0	Code
<code>childType</code>	<code>XmFRAME_TITLE_CHILD</code>	Code
<code>childVerticalAlignment</code>	<code>XmALIGNMENT_WIDGET_BOTTOM</code>	Code

5. Add an `XmTextField` to `frame`.
6. Set the following resources for `textField`:

Resource	Value	Location
<code>childHorizontalSpacing</code>	0	Code
<code>childType</code>	<code>XmFRAME_WORKAREA_CHILD</code>	Code
<code>childVerticalAlignment</code>	<code>XmALIGNMENT_WIDGET_BOTTOM</code>	Code

Status Check!

At this point, your user interface and widget hierarchy should look like the examples below. If they do, now would be a good time to save your work.



Copying an object

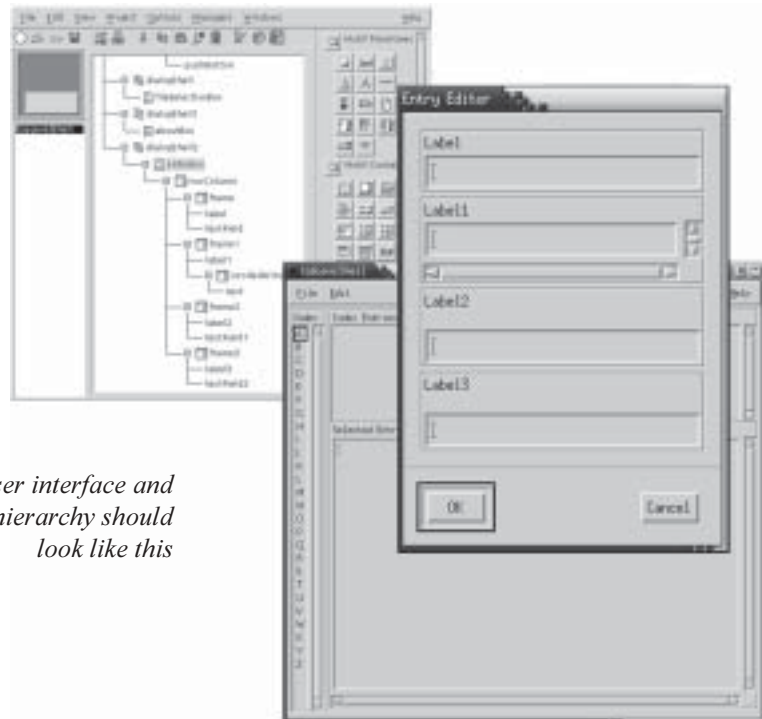
The following is best done using the Browser:

Adding a second frame

1. Place a second XmFrame (frame1) on the rowColumn in the interface.
2. Copy label1 into frame1. Select label1 with Ctrl/MB2, drag it to frame1. The Ctrl modifier causes Builder Xcessory to make a copy rather than reparent the object.
3. Add an XmScrolledText to frame1.
4. Add two more XmFrames to rowColumn by copying frame twice.

Select Ctrl/MB2, and drag it to rowColumn. This retains all the constraints and resources of frame and its children.

Since the last two items have the same structure as the “address” entry area (entryFrame), you can reuse the work you did to format that one.



Your user interface and widget hierarchy should look like this

Note: You may have to use your window manager to adjust the size of the Entry Editor to achieve the layout in the previous illustration

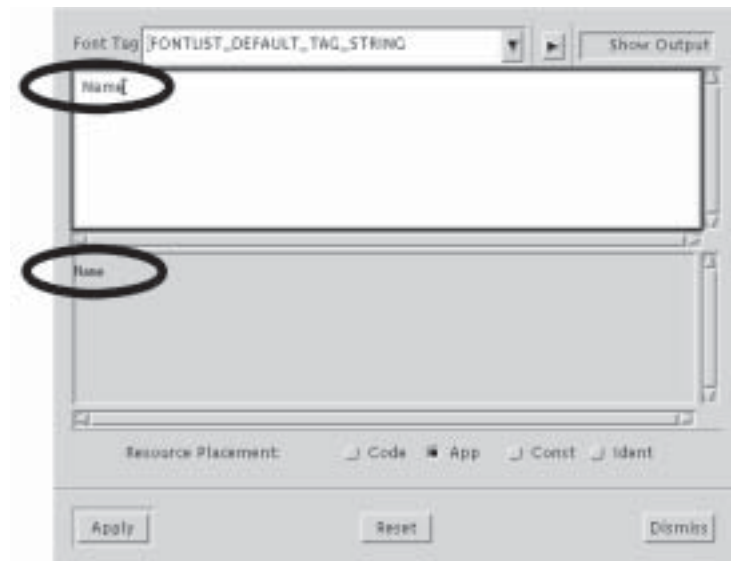
Labeling the Entry Editor Dialog

Label the four entry areas in the address Entry Editor by using an extended editor, as follows:

Using extended editor

Note: Using an extended editor allows you to set a resource in several objects without updating the Resource Editor.

1. Select `label` in the Browser.
2. Open the extended editor for the `labelString` resource.
3. Enter the label `NAME`. Click on the entry area, and then type in `NAME` to replace the previous data (in this case the default text “label”).
4. Select the **App** resource placement option.
5. Click on **Apply**. Leave the extended editor open.



6. Repeat steps 1–5 to set the `labelString` resource for the following objects:

Object	labelString	Location
label4	Address	App
label5	Phone	App
label6	E-Mail	App

7. Dismiss the extended editor.

Adding Control Buttons

To add control buttons to the address Entry Editor dialog, perform the following steps:

1. Select `entryBox` in the Browser.
2. In the Resource Editor window, click on the text `entryBox` to the right of the label “Instance Name”
3. Type the new name `entryEditor`.
4. Modify the following resources:

Resource	Value	Location
<code>cancelLabelString</code>	Cancel	App
<code>okLabelString</code>	OK	App

Note: Entering text in the `cancelLabelString` and `okLabelString` resources automatically adds the respective buttons to the interface.

Final Formatting of Entry Editor

To change the title of the Entry Editor, perform the following steps:

Changing Entry Editor title

1. Select `entryEditor` in the Browser.
2. Set the `dialogTitle` resource to `Entry Editor`, with **App** resource placement.

Adjusting Entry Editor Layout

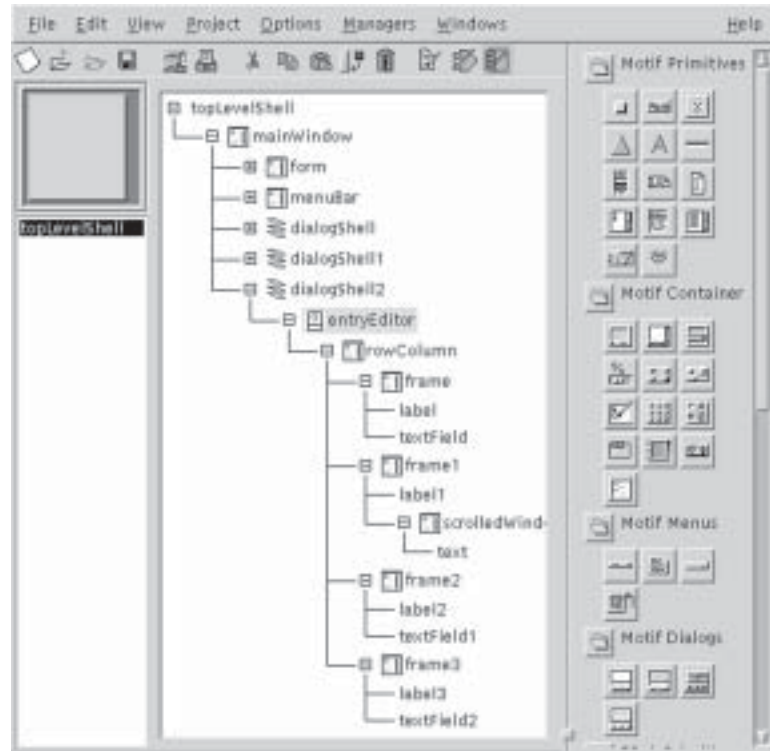
To do final adjustment of the layout of the address entry editor, perform the following steps:

1. Select `entryEditor` and all its children in the Browser by clicking on `entryEditor` with Shift/MB1. This selects the immediate item and all its children.
2. Select **Natural Size** on the Main Window **Edit** menu. This allows all the items in the address entry editor to fit together smoothly.

Hint: If you are resizing a large number of objects, it can be quicker to hide the objects before you do the resizing, since Builder Xcessory does not have to modify the screen for each change. To hide a group of objects, click on the folder icon by the parent object in the Browser tree.

The address Entry Editor should now appear like the following figure:





Saving the Interface

Select **Save** from the **File** menu in the Main Window, then click OK.

This writes out the file `u1l.uil`, which can be read back into Builder Xcessory to reconstruct the collection. You can also rename the UIL file at this point, for example, to `tutorial_4.uil`.

Summary

You have now completed Tutorial Four. In this tutorial, you have:

- Added a menu to your interface
- Added items to a menu
- Labeled menu items
- Created mnemonics for menu items
- Used Keep Parent to add multiple children to a widget
- Added a dialog to your interface
- Customized a standard dialog box

