Tutorial Three: 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 Six and Seven respectively.

Before You Begin This Tutorial

- Make a new directory called `tutorial_3` and change directories to `tutorial_3`.

Start Builder Xcessory

- If you have not started Builder Xcessory yet, enter the following at the prompt in the `tutorial_3` directory:

```
% bx50
```

Getting Started

Initialize your application as follows:

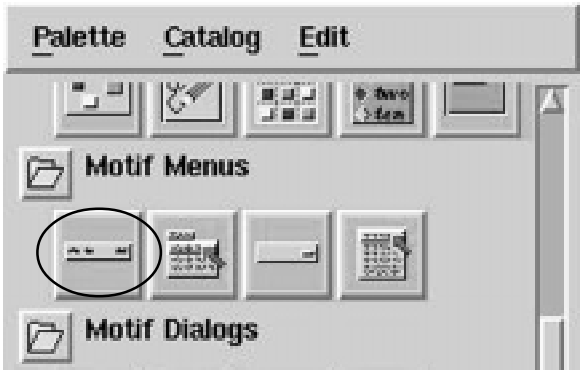
1. Clear the Builder Xcessory display by selecting **New** from the Browser **File** menu.
2. Open the UIL file you saved at the end of Tutorial 2:

```
<tutorial_path>/tutorial_2.uil
```

Creating a Menu Bar

Before you create the pulldown menus, you must add a MenuBar 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 MenuBar 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.

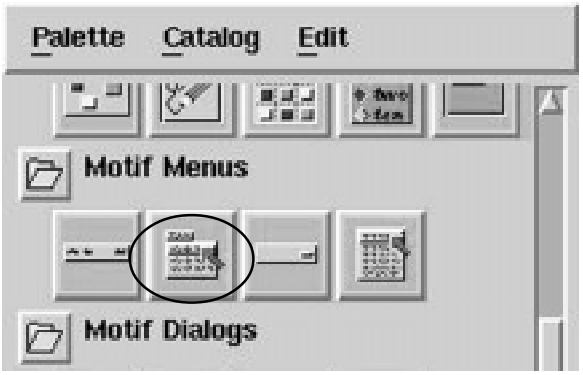
Creating Menus

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

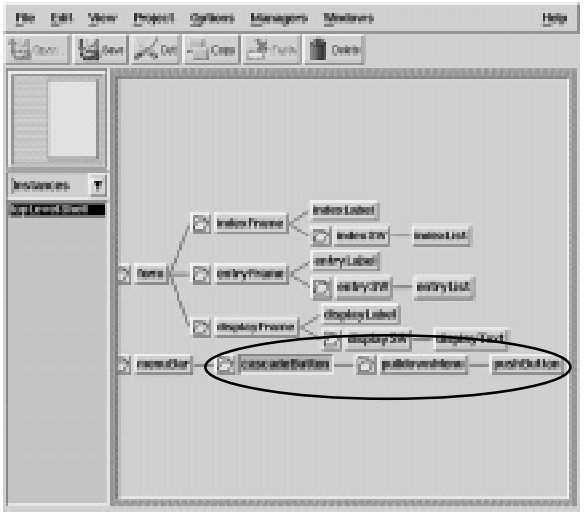
Adding a PulldownMenu

To create the File menu, perform the following steps:

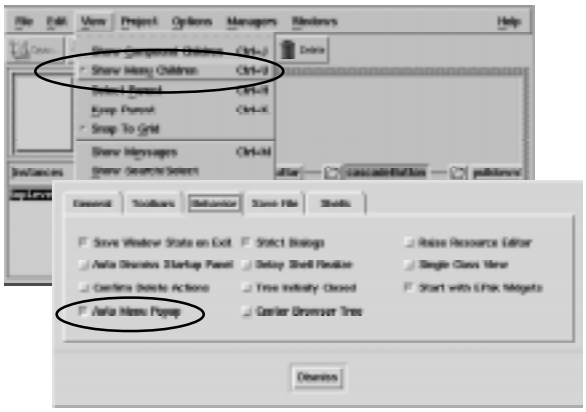
1. With MB2, select the XmPulldownMenu from the Palette and drop it onto the MenuBar in the interface.



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.



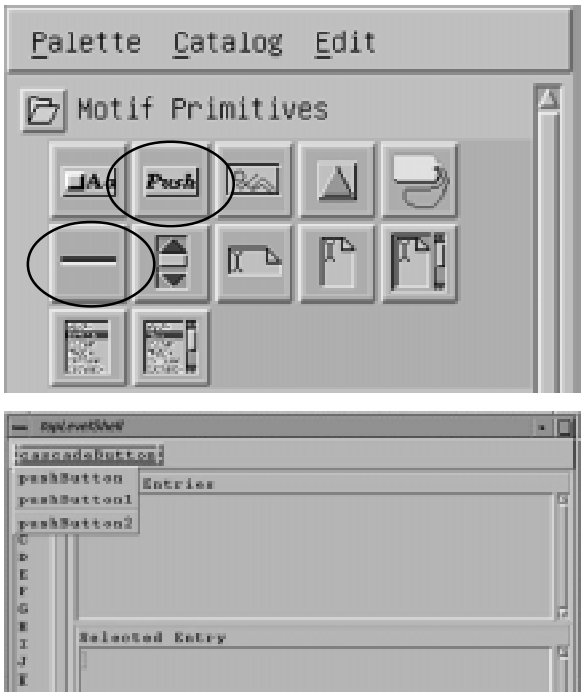
If you have set the User Preference Auto Menu Popup, the menu initially appears posted (i.e., “popped-up”). To unpost the menu, unset **Show Menu Children** in the Browser **View** menu or press Ctrl/U, the keyboard accelerator for that menu item. To change this default behavior, unset the **Auto Menu Popup** toggle in the **Behavior** tab of the **User Preferences** dialog. The **User Preferences** dialog is accessed via the **Options** menu choice in the Browser.



Adding Menu Items

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

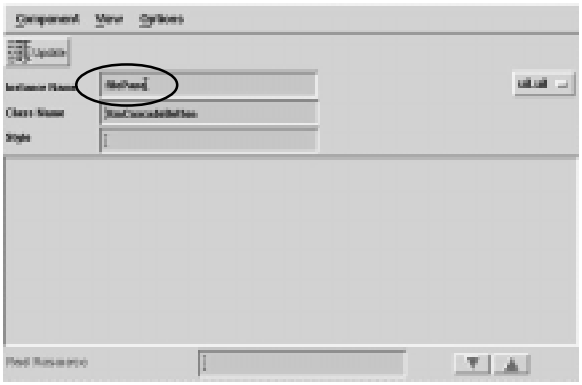
1. Select `cascadeButton` by clicking MB1 on the node in the Browser tree.
2. If `pulldownMenu` is not already posted, post it by pressing Ctrl/U.
3. Use MB2 to drag an XmPushButton to the `pulldownMenu` on the interface. Builder Xcessory automatically adds items you place on a menu at the bottom of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



User Interface After Adding Additional XmPushButtons and a XmSeparator

Naming widget instances

6. Select `cascadeButton` on the Browser.
7. Update the Resource Editor.
8. Change the Instance Name to `filePane`.



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

Object	New Name
pushButton	fileOpen
pushButton1	fileSave
pushButton2	fileExit

Labeling File Menu Components

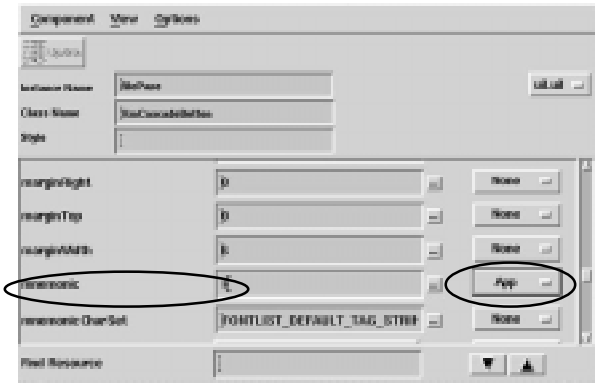
Now you will use the Resource Editor to add labels, accelerators, and mnemonics to the File menu. To label the File menu header, perform the following steps:

1. Select `filePane`.
2. Update the Resource Editor.
3. 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

5. Similarly, set the following resources for `fileSave`:

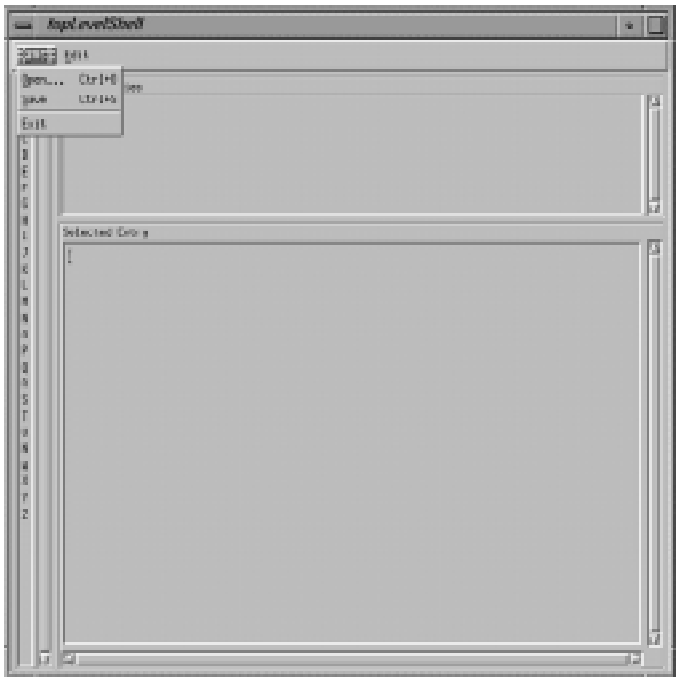
Resource	Value	Location
accelerator	Ctrl<key>S	App
acceleratorText	Ctrl+S	App
labelString	Save	App
mnemonic	S	App

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_3` directory before proceeding!)

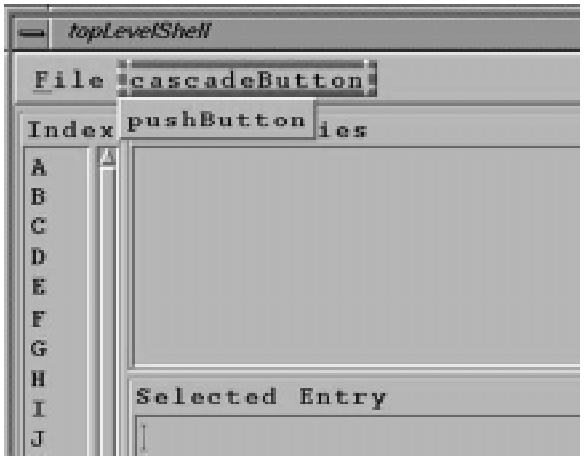


7. Select `filePane` again and hide the menu by pressing Ctrl/U.

Adding the Edit 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.

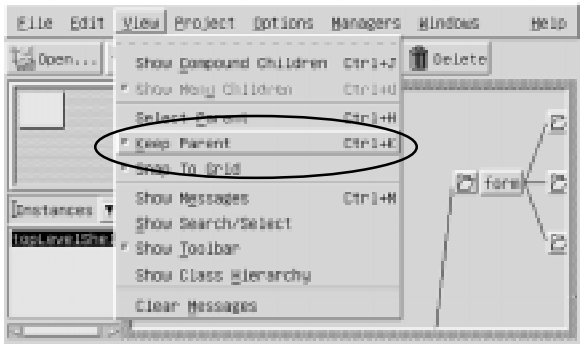


Using Keep Parent

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

1. Select the **Keep Parent** toggle from the Browser **View** menu.

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



Adding menu items

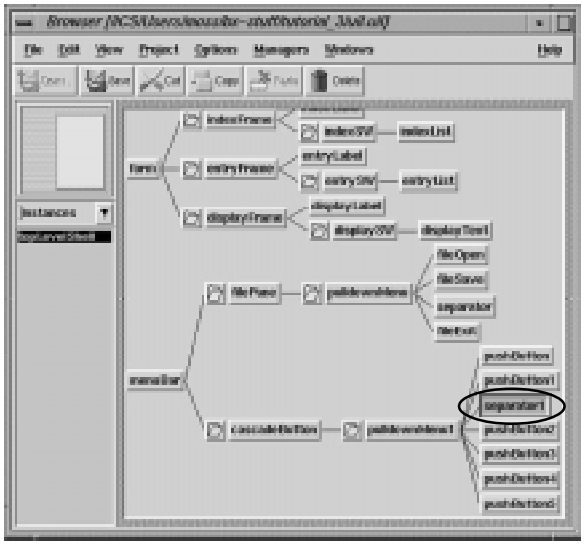
2. On the Browser, select `pulldownMenu1`.
3. Add five XmPushButtons.
   Because **Keep Parent** is on, you can add items to the interface simply by double-clicking on them in the Palette with MB1. Builder Xcessory knows to parent the new object to the selected object.

4. Add an XmSeparator to the interface.
   Double-click on XmSeparator on the Palette.

Moving a menu item

The Separator is at the bottom of the menu. To move it up to the correct spot, perform the following steps:

1. Select `separator1` on the Browser tree.
2. Press Ctrl/R to raise the Separator one level. (Ctrl/L lowers the item one level.)
3. Perform the raise operation three more times. `separator1` should now follow `pushButton1`.
4. Turn off **Keep Parent**.
   Unset the **Keep Parent** toggle on the Browser **View** menu.



Setting Instance Names for EditMenu 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 EditMenu.

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

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

1. Double-click on the instance name in the Browser to select it and update the Resource Editor. For example, click on `cascadeButton` in the Browser window to select it.
2. Then in the Resource Editor window, double-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 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 `cascadeButton` in the Browser.
2. Update the Resource Editor.
3. Change the following attributes and resources:

Resource	Value	Location
labelString	Edit	App
mnemonic	E	App

4. Similarly, set the following attributes for `pushButton` under `pulldownMenu1`:

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

5. Set the following attributes for `pushButton1` under `pulldownMenu1`:

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

7. Set the following attributes for `pushButton2` 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*.

8. Set the following attributes for `pushButton3` under `pulldownMenu1`:

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

9. Set the following attributes for `pushButton4` under `pulldownMenu1`:

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

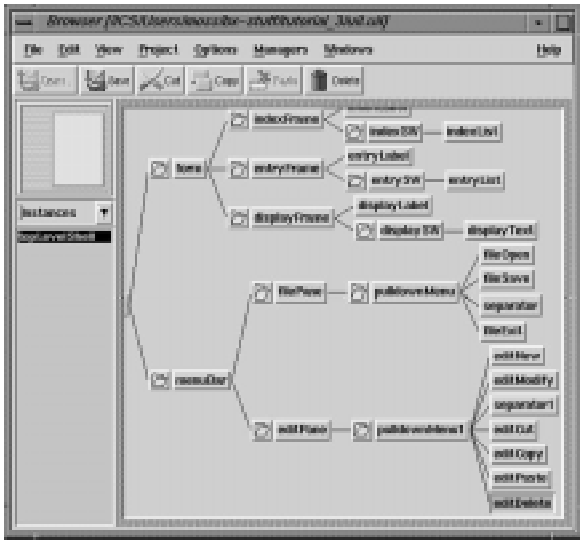
10. Set the following attributes for `pushButton5` under `pulldownMenu1`:

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 on the next page. If they match, save your work and proceed!



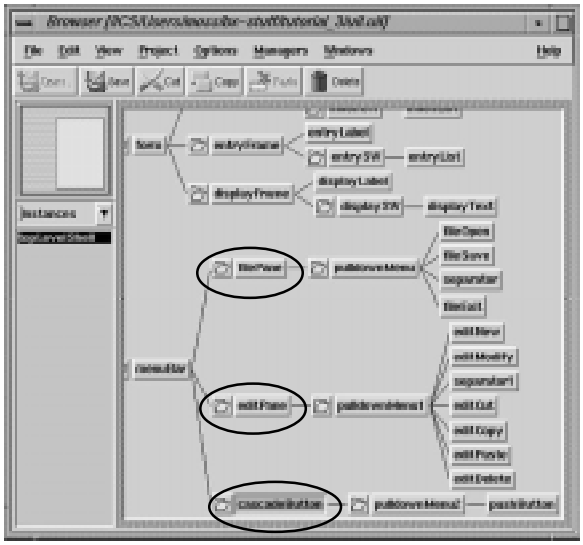
Structure of User Interface after Setting Resources for the Edit menu

Adding the Help Menu

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

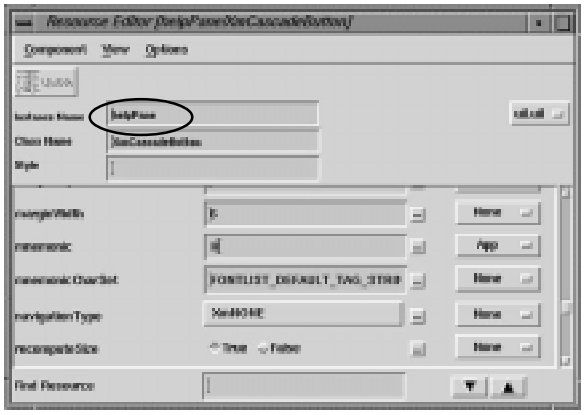
1. Add an XmPulldownMenu to the interface by dragging it to the MenuBar using MB2.

   Three objects—`cascadeButton`, `pulldownMenu2`, and `pushButton`—appear on the Browser tree under `menuBar`.



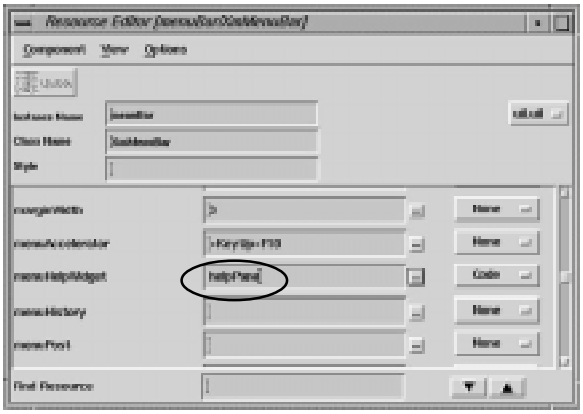
2. Select `cascadeButton` in the Browser (if not already selected).
3. Update the Resource Editor.
4. In the Resource Editor window, double-click on the text `cascadeButton` to the right of the label “Instance Name”

3. Type the new name `helpPane`.
4. Click on the check mark at the end of the text field, or press Return.



Setting Help button

3. Select `menuBar` in the Browser.
4. 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 MenuBar.

Labeling Help Menu Components

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

1. Select `helpPane` in the Browser.
2. Update the Resource Editor.
3. Change the following resources:

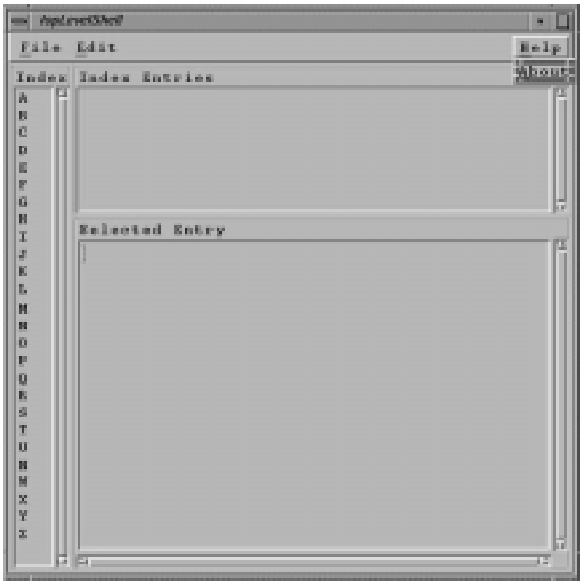
Resource	Value	Location
labelString	Help	App
mnemonic	H	App

4. 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.



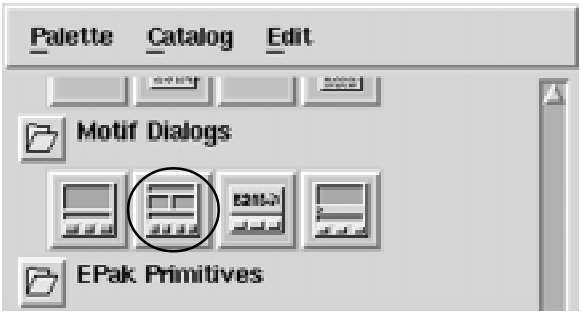
Creating Dialogs

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 use a FileSelectionBox.

1. Confirm that you still have **Keep Parent** selected by clicking on the **View** menu in the Browser 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.

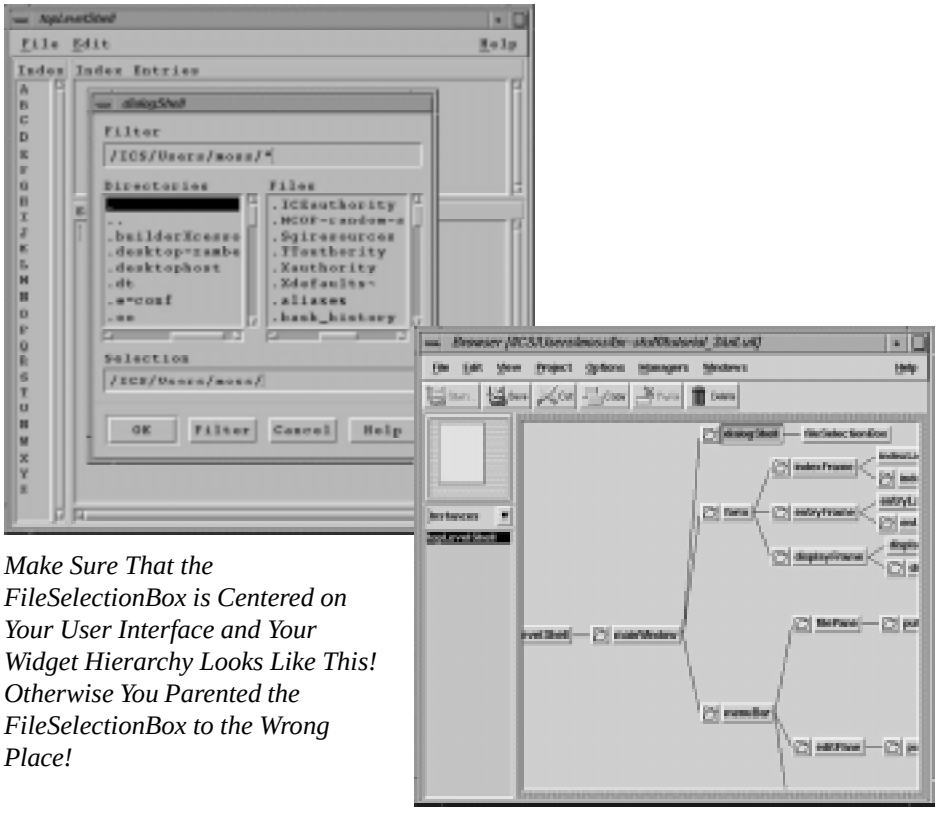
Note: Understand that you are using a two button mouse combination here. You click and release on the XmFileSelectionBox with MB1 and then move the outline over your user interface. Then you use the Ctrl/MB3 combination to drop it onto your user interface.

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.

Note: You can only use this technique to drag an object to the actual interface, not to the Browser tree (as you can with MB2).

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

Warning: If you have not set **Keep Parent** and are not quite accurate in where you drop the FileSelectionBox, it can 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.

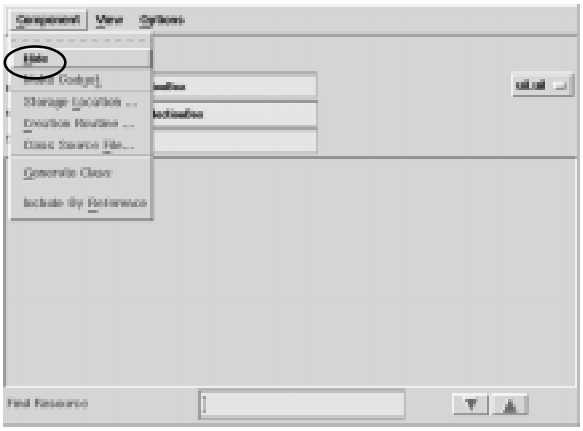


Make Sure That the FileSelectionBox is Centered on Your User Interface and Your Widget Hierarchy Looks Like This! Otherwise You Parented the FileSelectionBox to the Wrong Place!

Hiding an object

To hide `fileSelectionBox`, perform the following steps:

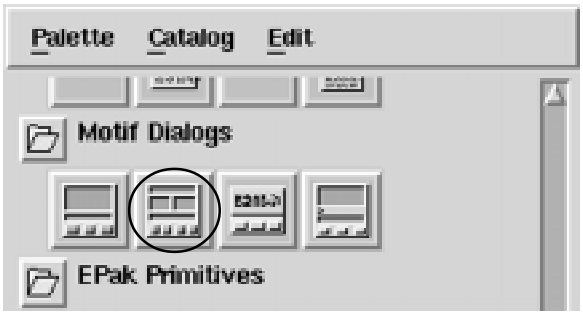
1. Select the `fileSelectionBox`.
2. Select **Hide** on the Resource Editor **Component** menu.



Adding a Message Box

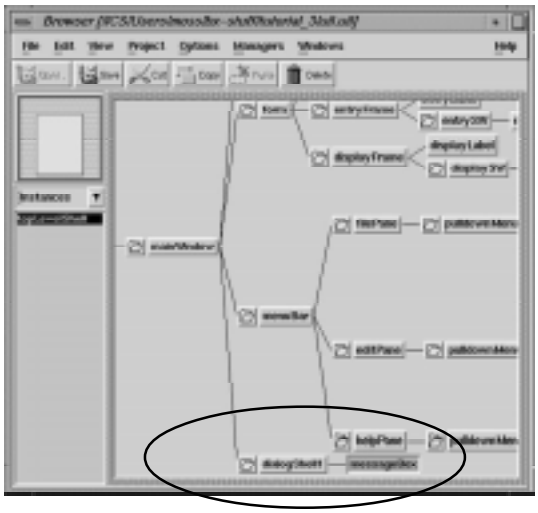
To add an "About" dialog for the Help menu, perform the following steps:

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.



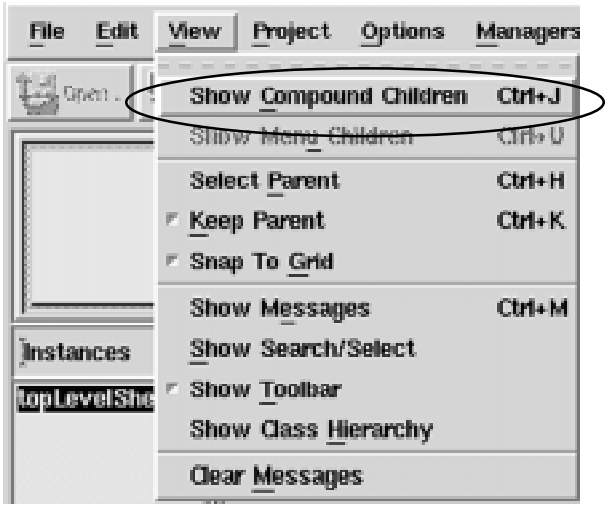
4. Move the outline of the XmMessageBox to the interface.
5. Drop the XmMessageBox anywhere on your user interface using the Ctrl/MB3 combination as in the previous section. The XmMessageBox is named `messageBox` in the Browser tree.

4. Compare your widget hierarchy with the one on the right to confirm that the XmMessageBox was properly parented.

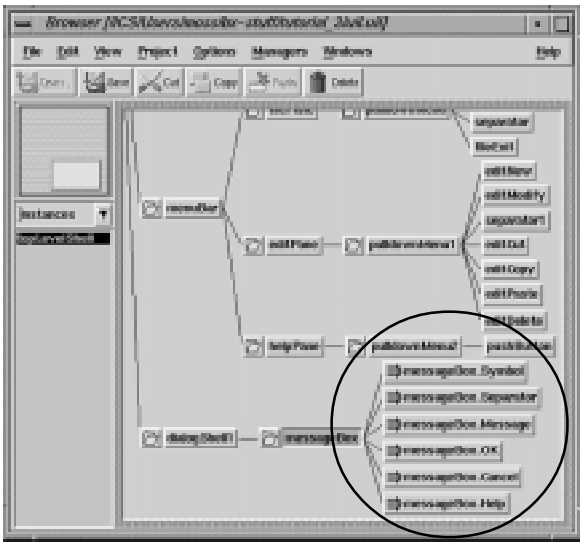


Viewing compound children

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 Browser **View** menu. The Browser tree now displays the children of all compound objects in the interface.



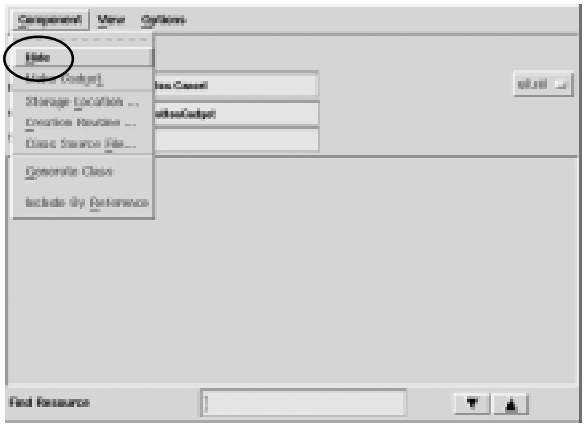
Note: Children of a compound object are indicated by a red arrow on the Browser tree. Builder Xcessory allows only limited modification of these children.



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. Update the Resource Editor.
3. Select **Hide** on the Resource Editor **Component** menu.
   The Cancel button disappears from the display and will not be included when you generate code.



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

Labeling the About Box

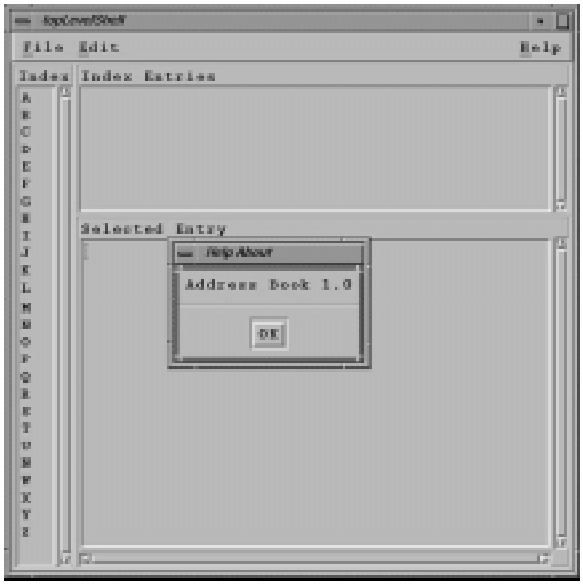
To add text to the about box, set the following attributes:

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

Resource	Value	Location
dialogTitle	Help About	App
messageString	Address Book 1.0	App
okLabelString	OK	App

.

The about box should now appear something like the figure below.



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:

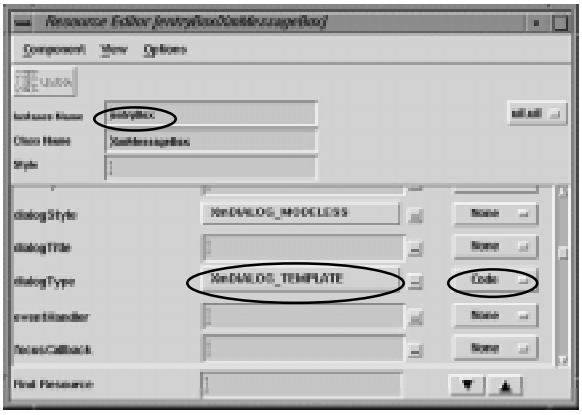
1. Click on XmMessageBox in the Palette with MB1.
2. Use Ctrl/MB3 to place the MessageBox in the interface. (Remember to make sure **Keep Parent** is selected and that you have `mainWindow` selected in the Browser!) This MessageBox is named `messageBox` on the Browser tree.

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



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

Adding first frame

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. Place an XmRowColumn in `entryBox`.
2. Place an XmFrame in `rowColumn`.
3. 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`:

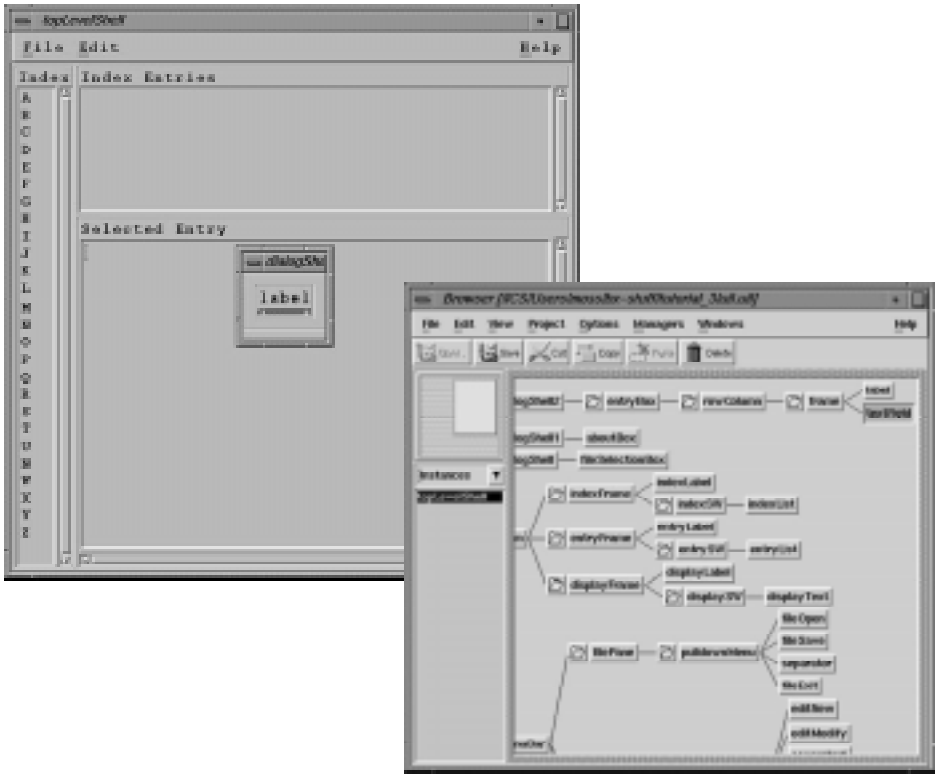
Resource	Value	Location
childHorizontalSpacing	0	Code
childType	XmFRAME_TITLE_CHILD	Code
childVerticalAlignment	XmALIGNMENT_WIDGET_BOTTOM	Code

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

Resource	Value	Location
childHorizontalSpacing	0	Code
childType	XmFRAME_WORKAREA_CHILD	Code
childVerticalAlignment	XmALIGNMENT_WIDGET_BOTTOM	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.



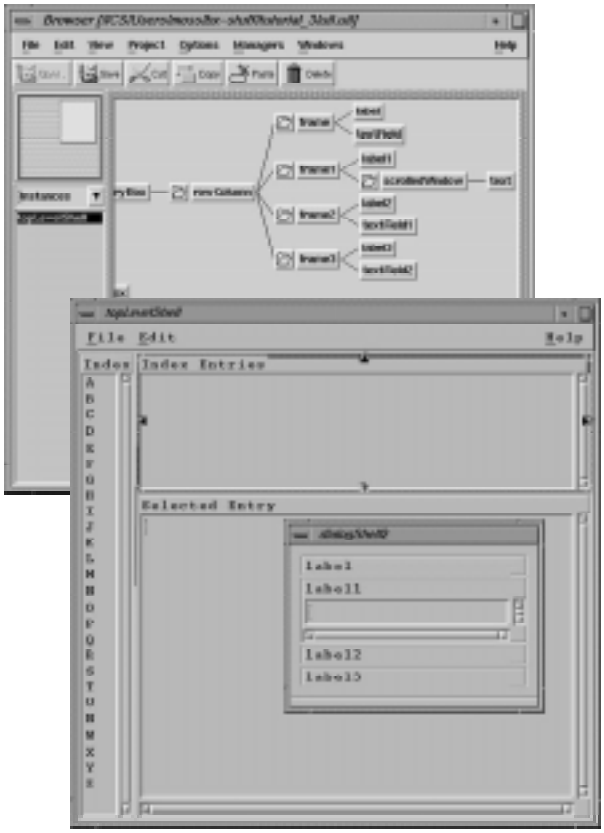
Copying an Object
Adding a second frame

1. Place a second XmFrame (`frame1`) on the `rowColumn` in the interface.
2. Copy `label` into `frame1`.
   Click on `label` 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.

   Click on `frame` with 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 the examples on the following page.



Labeling the Entry Editor Dialog

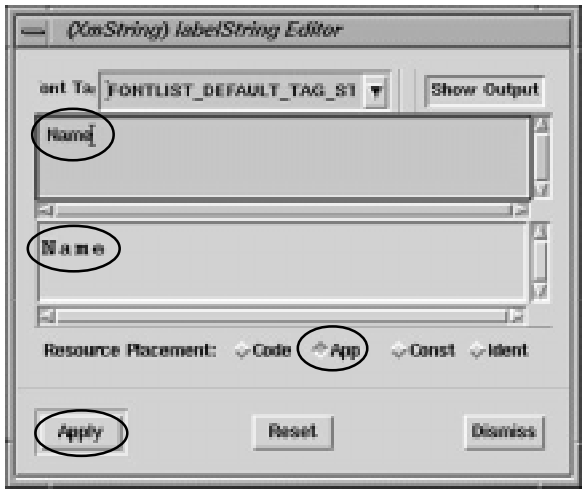
Using extended editor

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

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. Update the Resource Editor.
3. Open the extended editor for the labelString resource.

4. Enter the label “Name.”
   Double-click on the entry area, and then type in “Name” to replace the previous data (in this case the default text “label”).
5. Select the App resource placement option.
6. Click on Apply. Leave the extended editor open.



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

Object	labelString	Location
label1	Address	App
label2	Phone	App
label3	E-Mail	App

8. Dismiss the extended editor.

Adding Control Buttons

To add control buttons to the editor dialog, perform the following steps:

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

Resource	Value	Location
cancelLabelString	Cancel	App
okLabelString	OK	App

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

Final Formatting of Entry Editor

Changing Entry Editor title

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

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

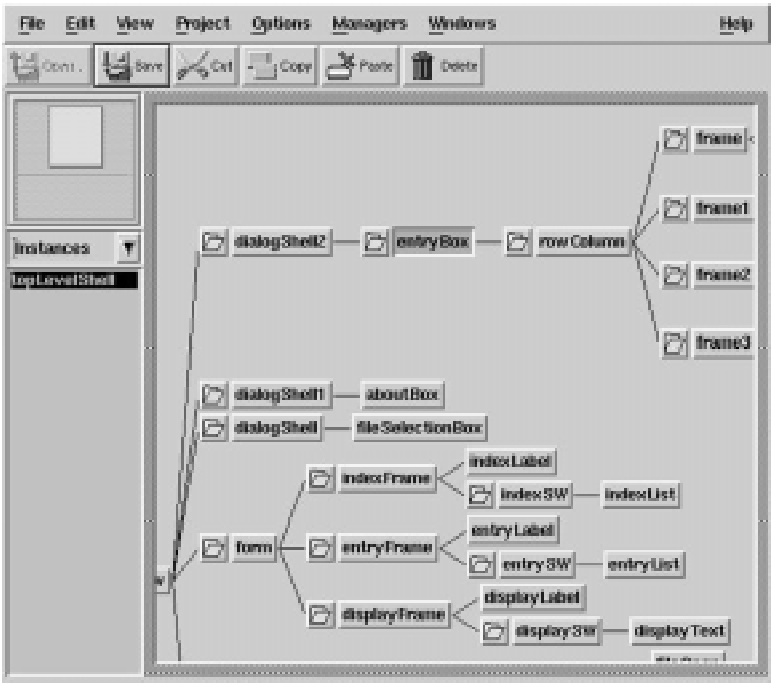
Adjusting Entry Editor layout

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 Browser **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:



Your Widget Hierarchy Should Look Similar to the Above Example

Saving the Interface

Select **Save** from the **File** menu on the Browser, then click OK.
This writes out the file `uil.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_3.uil`.

Summary

You have now completed Tutorial Three. 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