ICS Tutorials: Getting Started

Overview

The following chapters provide a series of tutorials to help you experiment with Builder Xcessory by creating a series of progressively more complicated interfaces. By working through the tutorials, you will learn:

- Basic Builder Xcessory features
- How to use Builder Xcessory to create well-formed Motif applications
- How to use Builder Xcessory to create reusable code, components, collections

The tutorials fall into three groups: 1–2, 3–6, and 7–8.

General Introduction

Tutorial One: Getting Started

This tutorial.

Tutorial Two: Basic Operations:

Illustrates the basic functions of Builder Xcessory’s two primary windows: the Main Window and the Resource Editor.

Creating and Customizing an Example

Tutorial Three: Manipulating Basic Objects

Demonstrates how to create and manipulate simple objects.

Tutorial Four: Menus and Dialogs

Demonstrates how to create and implement dialogs and menus.

Tutorial Five: Designing for Consistency and Flexibility

Shows how to create consistent interfaces by using resource styles, constants, and identifiers.

Tutorial Six: Designing for Reuse and Adding Behavior

Introduces classes and class instance resources.

Advanced Topics

Tutorial Seven: Working with ViewKit and Using Subclasses

Shows how to create dialogs and menus with ViewKit and introduces subclassing.

Tutorial Eight: Working with Java AWT

Demonstrates the creation of a set of Java classes that can be used as an applet or application.

Prerequisite Knowledge

This document assumes that you are familiar with the X Window System and OSF/Motif. If you are developing with Java AWT and/or ViewKit objects, this document assumes that you are familiar with these toolkits.

Building an Interface

Creating an Interface

To build and test an interface with Builder Xcessory, use the following procedure:

1. Create the objects that comprise your interface.
2. Specify values for the widget resources, including the callback resources that control the functionality of the interface.
3. Test the appearance of the interface.
4. Save the interface and generate code.

Adding Functionality

5. Edit the output files, especially those containing the callbacks structures, in order to connect the functionality of the interface.
6. Compile the interface.
7. Test the functionality of the interface.

Using the Tutorials

Each tutorial demonstrates the use of Builder Xcessory in constructing an interface. We recommend that you do them in the order presented as each tutorial builds on the previous one. The later tutorials tend to be more complex, and take for granted procedures described in detail in the earlier tutorials.

In general, if you are uncertain about the functionality of Builder Xcessory, review the *Builder Xcessory Reference Manual*.

Starting a new interface

Setting Up Tutorial Directories

When Builder Xcessory writes a Callbacks file to a directory containing another Callbacks file with the same name, new callback stubs are simply appended to the old Callbacks file. Therefore, if you are starting a new interface, it is a good idea to do the following:

1. Create a new directory.
2. Change to the new directory.
3. Start BX in the new directory.

In each of the following tutorials, you will create an interface and generate code. To prevent these files from overwriting one another, create the subdirectories `tutorial_3, tutorial_4,...tutorial_8`. Because you will not be saving any code in tutorials one (this tutorial) and two, you do not need to create subdirectories with those names. Before you start Builder Xcessory for a particular tutorial, you should first change to the appropriate directory.

If you finish one tutorial and want to go on to the next one, you can switch target directories by either:

- Saving your new user interface (see **Save** under the **File** menu in the Main Window) in the directory belonging to the new tutorial, or
- Changing the destination directory using the **File Names** tab of the **Code Generation Preferences** dialog that can be selected from the **Options** menu in the Main Window.

Both of these options are illustrated in Tutorial One: *Basic Operations*.

Notation Conventions

This document uses the following notation conventions:

Lists

The following two types of lists present procedures:

1. Numbered lists present steps to perform in sequence.
- Bulleted lists present alternate methods.

Objects

Objects are indicated as follows:

- Palette collection names are capitalized words with intercaps:
  Form or PushButton
- Instance names are lowercase words with intercaps:
  form or pushButton
- Class names are capitalized words with intercaps:
  Test or MyClass

Menu Notation

To indicate menus and menu options, the following format is sometimes used: BX_window_name:menu_name:menu_item(or dialog_selection). For example, Main:File:Exit is the **Exit** selection from the **File** menu of the Main Window. If a menu name is used by itself in a sentence, it is Bolded. For example:

- Select **Save** from the **File** menu of the Main Window.

Text

- Literals such as file names, directory paths, code and text as typed on the command line are printed in Courier font:
  ```
  .bxrc6
  /usr/ics
  -gen JAVA
  ```
- Text preceded by a % denotes text to enter at the command line:
  ```
  % bx
  ```
- Book titles, chapter names, and section names appear in *italic*:
  Builder Xcessory Reference Manual
  “Updating the Resource Editor”
- The main windows of Builder Xcessory are capitalized as follows:
  Main Window
  Resource Editor

Definitions

These tutorials use the following terms:

Drop

Release the mouse button after positioning the mouse (and screen object) as desired. Typically follows a drag operation. The phrase, “drop on to” indicates a drag and drop operation. Use MB2 to perform a drag and drop operation, unless otherwise specified.

Enter

Type a new value and press the Enter key.

Gadget

A user interface object built upon the Xt Intrinsics (the X Toolkit). Similar to a widget, a gadget lacks certain data structures (such as a window) contained in a widget. The gadget maintains this data on the client side, rather than on the server, as does a widget. Although seldom used with today’s server performance levels, gadgets remain supported by BX.

{lang}

Specifies the currently selected language for code generation.

MB1, MB2 and MB3

Mouse buttons one, two, and three. Typically, MB1 is the left-most button, MB3, the right-most. On a two-button mouse, MB2 is most commonly emulated by pressing both buttons simultaneously. For actions described as “click,” assume MB1.

MB3 Quick Access menu

This menu is invoked by pressing MB3 while the mouse pointer is positioned over an object on the display. The contents of the menu depend on the type of object pointed to and the window in which you access the menu.

Object/UI object

A reusable software entity with a user interface (UI), or **Widget**

A user interface object built upon the Xt Intrinsics (the X Toolkit). Motif user interface objects are widgets.

Notation of Wrapped Code

In the code fragments presented below, code that is intended to be entered as a single line in your program sometimes wraps around to a second line in the documentation. Indenting the second and subsequent lines denotes wrapped text. For example, the following lines should be entered on one line in your program:

```
XmAnyCallbackStruct
   *acs=(XmAnyCallbackStruct*)call_data;
```

In addition, because long identifiers are common in Motif, some identifiers are hyphenated.

What if Something Goes Wrong?

As you progress through the Builder Xcessory tutorials, you might miss a step, misunderstand our instructions or accidently change resources on the wrong Motif widget. It is also possible that despite of our best efforts to validate these tutorials in all environments, that a particular operation might not work for you.

To minimize the amount of backtracking you have to do, we include plenty of “Status Checks” during the tutorial. At these points, you should compare your user interface and the widget hierarchy displayed in the Browser portion of the Main Window against the sample we have provided. If they match, then you should save your user interface, and move on to the next step. **If they don’t match, stop and review your steps!** Most likely you missed a step along the way. If you can’t figure out what went wrong, go back to your last saved user interface and start again.