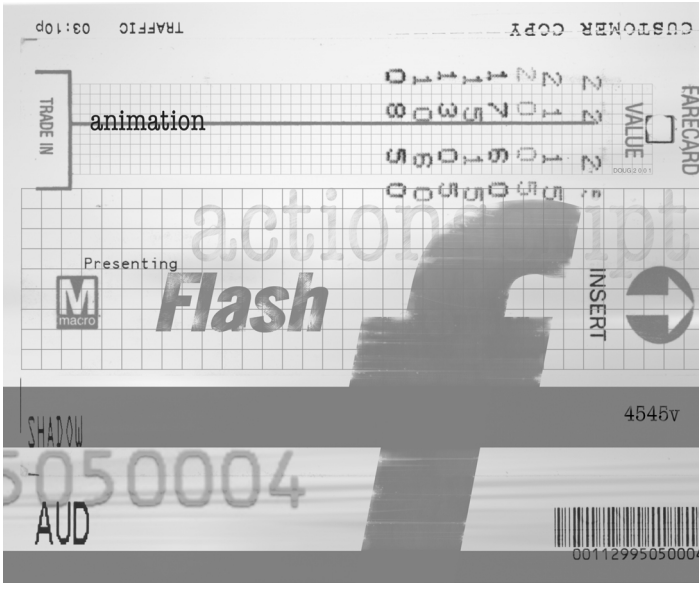
Chapter 22



Creating and Exchanging Your Own Smart Clips

You've added Smart Clips to movies, and you've modified them as well. Why not take the next step and build your own? All you need is an idea for a movie clip that would lend itself to Smart Clip style customization.

In this chapter, you learn how to do the following:

- **Create a Smart Clip.** You've already seen the power of Smart Clips. Now it's time to get busy and create one of your own.
- **Work with the Extension Manager.** You can download and install existing Smart Clips with the Extension Manager.
- **Package a Smart Clip for distribution.** Now that you've built a Smart Clip and know how to use the Extension Manager, try packaging your own Smart Clip to share with the world.

You'll begin by creating a brand new Smart Clip from scratch.

Creating Your Own Smart Clip

One possibility would be to create a Smart Clip that mimics a Microsoft PowerPoint presentation. How could you build a movie clip that would reveal a new bulleted point every time you press a key on the keyboard? The easiest way to approach this would be to embed a bullet point and a corresponding text field in a movie clip. You then can use the attachMovie method to add additional movie clips to the Stage.

Building a Base Clip

You need to get this concept working before you can convert it to a Smart Clip. The next exercise walks you through the complete process. Why make you complete the whole process? You'll never understand the distinction between a movie clip and a Smart Clip if you don't build a Smart Clip from scratch. This is going to take a while, so go get some coffee now.

Exercise 22.1 Creating the Base Movie for the Smart Clip

The first step is to create a movie clip with dynamic text fields for the bullet and the text.

1. Open a new file and create a new movie clip (Control+ or Command+F8) named **bulletFields**. This movie clip will contain two dynamic text fields: one for the bullet and one for the text.

2. Start with the text field for the bullet. Select the Text tool and open the Character panel. Change the following settings:

   Font: **Arial or Helvetica**
   Size: **24**
   Bold: **Selected**
   Italic: **Unselected**
   Color: **Pick one you like**
   URL: **Leave blank**

3. Draw a text box on the Stage. This will be your bullet point, so it needs to hold only one character of information. Make it a little wider than you might think you need. This leaves room for the bolding.
4. Switch to the Text Options panel and make the following changes:

   Text Type: **Dynamic Text**
   Text Line: **Single Line**
   Variable: **bulletType**
   HTML: **Not selected**
   Border/Bg: **Not selected**
   Selectable: **Not selected**

5. Position the text box so that the upper-left corner of the text box is centered on the movie clip's Registration point.
6. Draw a second text box to the right of the first text box. This one should be large enough to hold a long line of text.
7. In the Text Options panel, make the following changes for the second text box:

   Variable: **bulletText**
   HTML: **Selected**

8. Switch to the Character panel and turn bolding off.
9. Align the text boxes with the bulletText to the right of the bulletType. (See Figure 22.1.)
10. To use the attachMovie method, you need to set up the linkage properties for the bulletFields movie clip. Open the Library and select the bulletFields movie clip. Click the Options triangle and select Linkage from the pop-up list.
11. In the Symbol Linkage Properties pane, click the Export This Symbol Option button, type **bulletFields** in the Identifier field, and then click OK. (See Figure 22.2.)

    Having the Linkage Identifier and the movie clip use the same name helps prevent confusion down the road; you don't have to remember what identifier you assigned to the movie clip because it's the same as the movie clip name.

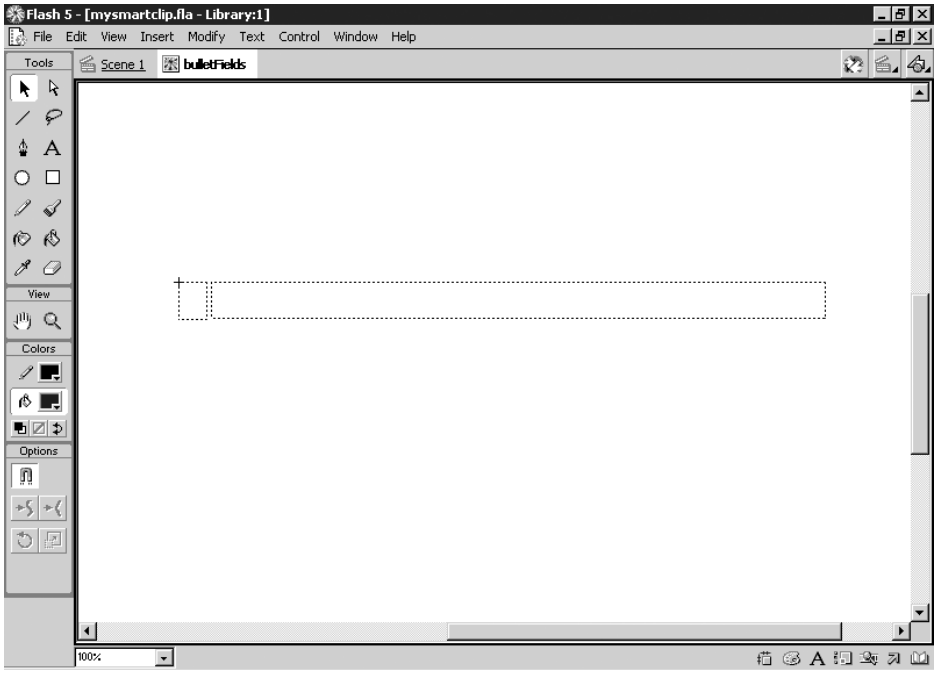


Figure 22.1 Align the text box for the bullet point so that its upper-left corner is aligned with the Registration point. The text box for the bullet items is positioned to the right of the first text box.

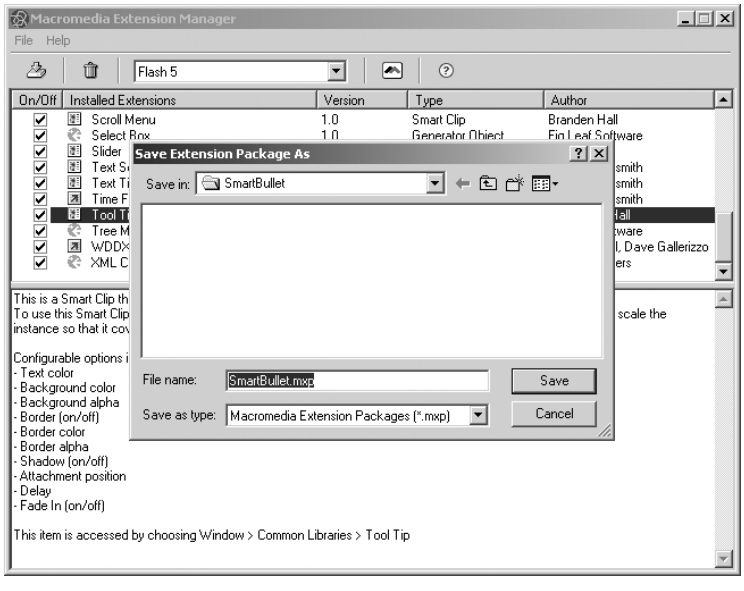


Figure 22.2 Before you can use the attachMovie method, you have to set up the linkage properties for the movie clip.

Next you'll embed the bulletFields movie inside a placeholder movie clip on the main timeline. The placeholder movie clip will contain the bulletFields and all the ActionScript that controls the movie. Why not have the code on the main timeline rather than on a movie clip? You do so because you need to use the onClipEvent method, and that has to be attached to a movie clip. You also know you are going to package this as a Smart Clip, which also means you can't use the main timeline.

12. Return to the main timeline, create a new movie clip symbol (Control+ or Command+F8), and name it **Controller**.
13. Inside the new Controller movie clip, rename layer 1 **bulletFields** and drag a copy of the bulletFields movie clip from the Library onto the Stage.
14. Center the upper-left corner of the bulletFields movie clip with the Registration point of the Stage.

Note

Why do we always align the upper-left corner of the movie clip content with the movie clip's Registration point? It makes positioning the elements on the Stage a lot easier. Remember that the Flash movie's (0,0) position is the upper-left corner. If your movie clips are oriented in the same way, it's a snap to figure out how to place them on the Stage.

15. Return to the main timeline and drag a copy of the Controller movie clip from the Library onto the Stage.
16. Save your movie as **SmartBullet.fla**. Go ahead and test your movie now. You shouldn't see anything. The text fields in the bulletFields movie clip are empty.

Next you'll set up an array to hold the information you want to display for your bullet points. First you'll create the new Array object, and then you'll populate it.

Exercise 22.2 Setting Up and Populating the Array

You'll use an array inside an onClipEvent(load) to hold the information for each line you want to have appear in your movie. You use the load clip event because you want the information to load the first time this movie clip appears on the Stage, and only then.

1. Open SmartBullet.fla, if it isn't still open, or open SmartBullet.fla from the Chapter_22/Assets folder on the CD.
2. Select the Controller movie clip on the Stage and launch the Actions panel.

3. Creating and populating the array is fairly straightforward. You'll spice things up a bit by adding some HTML formatting. Select the instance of the Controller movie clip on the Stage and add the following code to the Actions list:

```
onClipEvent(load) {
    textItems = new Array();
    textItems[0] = "<b>This is line 1.</b>";
    textItems[1] = "<i>This is line 2.</i>";
    textItems[2] = "<u>This is line 3.</u>";
    textItems[3] = "This is line 4.";
}
```

   You've got your items in an array. You know you're going to be displaying them one at a time. That means you're going to need a way to keep track of where you are in the array. You'll also need to know how long the array is. (Yes, I know you know it has four elements, but that might change.) You need to set up a variable—currentItem—to track your array and then use the array.length property to check your array to see how long it is.

4. Add the code that's between the asterisks:

```
onClipEvent(load) {
    textItems = new Array();
    textItems[0] = "<b>This is line 1.</b>";
    textItems[1] = "<i>This is line 2.</i>";
    textItems[2] = "<u>This is line 3.</u>";
    textItems[3] = "This is line 4.";
//****************************************
    currentItem = 0;
    totalItems = textItems.length;
//****************************************
}
```

5. There's just a bit more housekeeping to do inside this onClipEvent. You need to decide what character you are going to use for a bullet point, and you need to set a variable that will control how much space to put between the movie clips as they are attached to the Stage. After the last line of code you just entered, but before the curly brace, add the following code:

```
bulletType="*";
itemSpace=30;
```

6. Save your movie.

   Your code at this stage should look like Listing 22.1. Make sure you've closed all your curly braces and have semicolons at the end of the statements.

Listing 22.1 What Your Code Should Look Like After Exercise 22.1

```
onClipEvent(load) {
    textItems = new Array();
    textItems[0] = "<b>This is line 1.</b>";
    textItems[1] = "<i>This is line 2.</i>";
    textItems[2] = "<u>This is line 3.</u>";
    textItems[3] = "This is line 4.";
    currentItem= 0;
    totalItems = textItems.length;
    bulletType = "*";
    itemSpace = 30;
}
```

You've finished setting up what will happen when the movie clip loads. Now you need to set up the events that will occur when you press a key on the keyboard.

Attaching the New Movie Clips to the Stage

What needs to happen in this section of the code? You need to set up a new onClipEvent to react to keyDown events. Every time a key gets pressed, you need to do several things:

1. Check whether the currentItem is less than the totalItems.
2. If not, you'll stop performing your Actions. If currentItem *is* less than totalItems, you'll attach a movie clip to the main timeline. Remember that whenever you start attaching movie clips, you need to assign them unique names and unique levels to occupy. You can use the value of currentItem to help you do this.
3. You need to be able to position each movie clip relative to the others. In this case, the X position will always be 0, but each subsequent movie clip's Y position will change by the value you set in itemSpace.
4. Finally, you'll need to increment the value of currentItem.

You can begin by setting up the onClipEvent and attaching the movie to the main timeline.

Exercise 22.3 Setting Up the onClipEvent(keypress)

You need to set up the onClipEvent(keypress) to get your movie clip to respond to input from the keyboard.

1. Open SmartBullet.fla, if it isn't still open, or open SmartBullet1.fla from the Chapter_22/Assets folder on the CD.
2. Make sure you have the Controller movie clip on the main timeline selected and the Actions panel open. After the last curly brace, you'll set up the shell for the onClipEvent. Enter the following code:

```
onClipEvent(keyDown){

}
```

3. On the blank line between the curly braces, you'll set up your if statement. Set up a loop to check if the currentItem is less than the totalItems. Then have that loop increment currentItem. Enter the code between the asterisks:

```
onClipEvent(keyDown){
//**************************************************
//check to see if there are any more items to display
    if (currentItem < totalItems){

    ++currentItem;
    }
//**************************************************
}
```

4. Now you just need to add in the actions that will occur after you enter the loop. You need to attach the movie clip, assign it a unique name and level, and then set the X and Y position for the movie clip. Enter the code between the asterisks:

```
onClipEvent(keyDown){
//check to see if there are any more items to display
    if (currentItem < totalItems){
//**********************************************************
// Attach the new movie clip
        _parent.attachMovie("bulletFields", "item"+currentItem,
        ➥currentItem);
//Set the x position of the attached movie clip
        _parent["item"+currentItem]._x = 0;
//Place the new item below the previous item
        _parent["item"+currentItem]._y = currentItem * itemSpace;
//Grab the bullet type
        _parent["item"+currentItem].bulletType = bulletType;
//Grab the bullet text
        _parent["item"+currentItem].bulletText =
        ➥[textItems[currentItem];
//**********************************************************
        ++currentItem;
    }
}
```

Notice that you are using array syntax to create dynamic variable names. Do you remember Chapter 14, "Introduction to Object-Oriented Programming?"

5. Now you can test your movie. Press any key on the keyboard to activate the onClipEvent(keydown), which will attach the next bulleted item.

Tip

Backspace, Tab, and Enter do not count as key presses!

If the movie clip works properly, you're ready to convert your plain old movie clip into a "smart" movie clip.

Tip

If the HTML tags are showing up in your bulleted list, you need to select the bulletText field inside the bulletFields movie clip and verify that you selected HTML on the Text Options panel.

Converting the Base Movie to a Smart Clip

The first thing you'll do is package the movie clip you just created inside another movie clip. Then you'll convert the new movie clip into a Smart Clip by defining its Clip Parameters.

Exercise 22.4 Converting Your Movie Clip to a Smart Clip

You'll begin by embedding your movie clip inside a new movie clip.

1. Open SmartBullet.fla, if it isn't still open, or open SmartBullet2.fla from the Chapter_22/Assets folder on the CD.
2. Select the Controller movie clip on the Stage and embed it in a new movie clip by choosing Insert > Convert to Symbol. Give the new symbol the name **smartBullets,** and click OK.
3. Double-click the smartBullets Smart Clip to open it in Symbol-Editing mode. Reposition the controller movie clip so its upper-left corner aligns with the Smart Clip's Registration point.
4. Open your Library and right- or Control-click the smartBullets movie clip, and then select Define Clip Parameters.
5. Click the plus [+] to add a parameter.
6. Double-click varName and change the parameter's name to **textItems.**
7. In the Type field, double-click Default to open the drop-down menu, and then change the type to Array().
8. Double-click the Value field for the array to open the Values panel.

9. To add a new value, click the plus (+) button and then double-click the defaultValue to change it. Add the following code:

```
<b>This is line 1.</b>
<i>This is line 2.</i>
<u>This is line 3.</u>
This is line 4.
```

10. Click OK. This should look familiar. These are the same values you pushed to your Array in the Controller movie clip.
11. Add another variable (+), and name it **bulletType**. Change the Type to List(), and then double-click its Value to enter several types of bullets, such as -, +, *, and >. Click OK.
12. Add one last variable, and name it **itemSpace**. Change the Value to 30, leave the Type set to default, and select OK to return to the Stage.

    Now you'll need to modify the code on the Controller movie clip so that instead of referring to its own internal variables, it refers to the variables that the user sets for each instance of the Smart Clip through the Clip Parameters panel.
13. Double-click the smartBullets movie clip to open it in Symbol-Editing mode. Select the Controller movie clip on the Stage and launch the Actions panel.
14. Inside the onClipEvent(load), delete the code that creates and populates the textItems[] array.
15. Change the value for totalItems so that it equals _parent.textItems.length.
16. Delete the code that sets the bulletType and itemSpace variables. Your code in onClipEvent(load) now will look like this:

```
onClipEvent(load){
    currentItem = 0;
    totalItems = _parent.textItems.length;
}
```

17. In the onClipEvent(keyDown), add **_parent.** in front of the itemSpace, bulletType, and textItems[] variables, so that the variables from the smartBullets Smart Clip instance are sent to the attached instances of the bulletFields movie clip. The code inside your onClipEvent(keyDown) now will look like this:

```
onClipEvent(keyDown){
    if (currentItem < totalItems){
        _parent.attachMovie("bulletFields", "item"+currentItem,
        ➥currentItem);
        _parent["item"+currentItem]._x = 0;
        _parent["item"+currentItem]._y = currentItem *
        ➥_parent.itemSpace;
        _parent["item"+currentItem].bulletType = _parent.bulletType;
        _parent["item"+currentItem].bulletText =
        ➥_parent.textItems[currentItem];
        ++currentItem;
    }
}
```

18. Test your movie, using any key to cause the smartBullets Smart Clip to dynamically attach instances of the bulletFields movie clip. Notice that the X and Y positions of the bulleted items now are dependent on where the Smart Clip is placed on the Stage. If you want to change the values being used, just edit the Clip Parameters.
19. You have one last bit of housekeeping to do. You need to create an assets folder in the Library and move the movie clips and graphics that support the Smart Clip into the folder. Why? It keeps your file organized, and if you ever plan to package your Smart Clip for distribution on the Flash Exchange, Macromedia will insist that you do so.
20. Open the Library and create a new folder called **smartBullet_assets**. Drag the controller and bulletFields movie clips into the new folder.
21. Save your file.

There you have it—your very own Smart Clip. Granted, this is a simple example, but now that you know the basics, you can modify it to do whatever you want. In fact, you already know that you can use text files as data sources for your Smart Clips. You could do that for this one as well. Refer back to Chapter 20, "Using Flash 5's Stock Smark Clips," if you need a refresher course. In the meantime, take a look at the Macromedia Flash Exchange and explore how you can submit your own Smart Clips to the Exchange.

The Macromedia Flash Exchange

The Macromedia Flash Exchange is a clearinghouse for extensions for Flash 5 that have been created either by Macromedia or by members of the Macromedia development community. Most of the extensions are free. The available extensions include Smart Clips, ActionScript files, mini-applications, tutorials, templates, Generator objects and templates, clip art, previous Macromedia Dashboard movies, and more. Before you start any project, visit the Flash Exchange. You don't want to waste your time reinventing the wheel.

Before you can use the extensions at the Flash Exchange, you need to download and install the Macromedia Extension Manager. With the Extension Manager installed, you can install or delete extensions for Flash 5, DreamWeaver 3, or UltraDev 4.

The really cool thing is that you can use the Extension Manager to package your own Smart Clips for inclusion on the Exchange. You can share your knowledge with the world. (See Figure 22.3.)

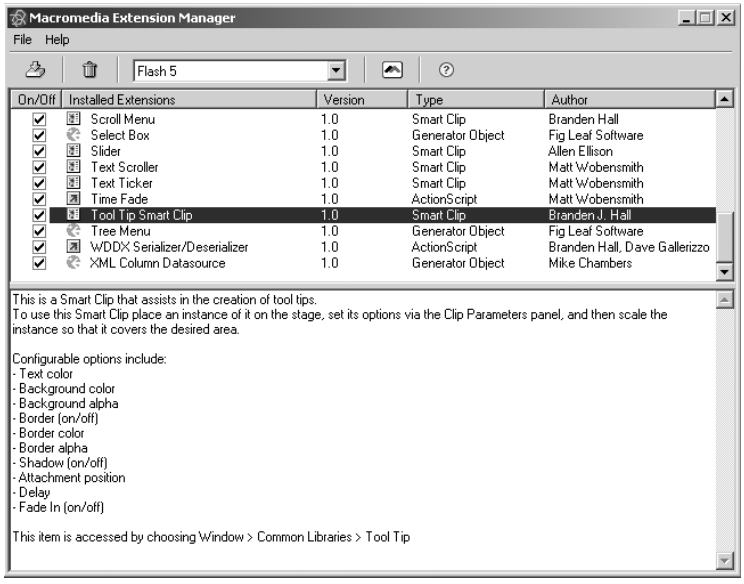


Figure 22.3 The Macromedia Extension Manager is used to take the pain out of installing extensions for Flash 5, Dreamweaver 4, and UltraDev 4. You also can use the Extension Manager to package your own extensions and submit them to the Flash Exchange.

The Macromedia Extension Manager

The Extension Manager is a free addition to Flash 5 that enables you to add, delete, and package extensions for Flash. To download the Extension Manager, go to `http://www.macromedia.com/exchange/flash` and find the Macromedia Extension Manager link. To install the Extension Manager, follow the directions on the download page.

After you install the Extension Manager, you can access it by choosing Help > Manage Extension Items from the main menu in Flash 5. Of course, the Extension Manager is no fun if you don't have anything to install, so you need to go back to the Flash Exchange and pick an extension or two to download.

If you want to take a look at a fun Smart Clip, do a search on "Tool Tip" and download Branden Hall's Tool Tip Smart Clip.

Notice the filename of the file you are downloading. It ends in *MXP*. MXP stands for Macromedia eXtension Package. The MXP file bundles two other files:

- The file(s) to be installed (FLA, AS, and so on)
- The Macromedia eXtension Installation (MXI) file used to handle the installation

You'll be learning more about MXI and MXP files later in this chapter. For now, just download an extension to your hard drive.

After you downloaded a file, you need to do the following:

1. Open the Extension Manager (Help > Manage Exchange Items).
2. Select File > Install Extension, browse to the MXP file you downloaded, and then click Install.

That's it. You don't even need to restart Flash. After the extension is installed, the name of the extension and its version, type, and author will appear in the upper pane of the Extension Manager.

To see a description of what the extension does and how you use it, just click the extension name to see a description in the lower pane.

To uninstall an extension:

1. Select the extension.
2. Choose File > Remove Extension (or click the Trash Can icon).
3. Click Yes to remove the extension.

If the removal was successful, you'll get a pop-up message that says "The extension has successfully been removed."

Now that you have the Extension Manager installed, why not use it to package the Smart Clip you just created?

Packaging an Extension

Packaging a Smart Clip for inclusion on the Flash Exchange is a six-step process:

1. Create an awesome Smart Clip.
2. Make sure you have the latest version of the Macromedia Extension Manager.
3. Create an installation file.
4. Package the extension.
5. Test your packaged extension to make sure you haven't made any bone-headed mistakes.
6. Submit your file to Macromedia so that they can test to make sure you haven't made any bone-headed mistakes.

After you've completed all these steps, you can sit back and win accolades and admiration from your peers. Or not.

Why not try packaging the Smart Clip you just created? I strongly suggest you skip Step 6—submitting the extension to Macromedia—unless you've made some significant performance enhancements to the existing clip.

Packaging Your Smart Clip

You've already knocked off the first requirement. So maybe it's not an awesome Smart Clip, but it's something to work with. Don't complain—you didn't have to think of it yourself.

Checking the Version of Your Extension Manager

The next thing to do is make sure you have the most recent version of the Extension Manager. If you just downloaded and installed the Extension Manager, skip to the next section. As of September 2001, the most recent version of the Extension Manager was Version 1.2. To check your version, do the following:

1. Visit the Exchange at `http://www.macromedia.com/exchange/Flash` and click the link to the Macromedia Extension Manager to see what the latest version is.
2. Open Flash 5.
3. Select Help > Manage Exchange Items from the main menu.
4. When the Extension Manager opens, select Help > About Macromedia Extension Manager.
5. The version number is in the lower-left corner.

If your version matches the one on the Macromedia site, you're good-to-go. If it doesn't, you need to download and install the new Extension Manager.

After you're sure you have the most recent version of the Extension Manager, you're ready for the next step.

Creating an Extension Installation File

The file that you need to create to install your extension is an Extensible Markup Language (XML) file that you save with an MXI file extension. Don't freak out. XML is just a standard format for exchanging information. It structures information in a way that is easily reused in multiple situations and it's not hard to work with at all. You'll be working more extensively with XML in Chapter 23, "Introduction to XML." In this chapter, you'll walk through the basics of creating an installation file for your Smart Clip. The MXI file extension just stands for Macromedia eXtension Installation.

What does the XML in the MXI file look like? It's not terribly complex, as you can see in Listing 22.2. The only things that you actually need to change are bolded.

Listing 22.2 The XML Inside the MXI Installation File Is Really Not Very Complex, and There Are Only a Few Attributes That You Need to Change

```
 <macromedia-package
 name="name of smart clip"
 version="1.0"
 type="Smart Clip">

<author name="your name" />

<products>
   <product name="Flash" version="5" primary="true" required="true" />
</products>

<description>
<![CDATA[
Describe the smart clip.
]]>
</description>

<ui-access>
<![CDATA[
Tell them where to find the smart clip
]]>
</ui-access>

<files>
  <file source="smart clip file name" destination="$flash/Libraries" />
</files>

</macromedia-package>
```

As you can see from the listing, the code has a very specific structure to it. The most common mistake you'll make when working with XML files is to not close one of the tags. Most of the tags come in pairs, such as the following:

```
<products></products>
```

The tags that don't come in pairs don't contain any additional content and must end with />, as in the following:

```
<author name="your name" />
```

For more detailed information on creating installation files for extensions other than Smart Clips, download a copy of the MXI file format PDF from Macromedia's Web site. You also can download the MXI sample files by Matt Wobensmith on the Flash Exchange.

These are MXI files that you can adapt for your own use. The MXI samples are available as an extension from the Flash Exchange. With these files as a template, you can quickly set up installation files for any kind of Exchange item. After you walk through the process for setting up one MXI file, the rest are easy. At the end of this chapter, you'll find some guidelines for setting up installation files for other types of extensions.

Now it's time for you to create an installation file for your Smart Clip.

Exercise 22.5 Creating an Installation File (MXI)

You'll start by modifying an existing template MXI file.

1. Open sample.mxi in a text editor, such as NotePad or SimpleText. You can find the file in the Chapter 22/Assets folder.
2. The first thing you need to do is give the Smart Clip a title. The title is how it will be listed on both the Flash Exchange and in the Extension Manager. In the very first tag in the file, change this:

   ```
   name="name of Smart Clip"
   ```

   To the following:

   ```
   name="Smart Bullet Points"
   ```

3. Next you need to tell the world who you are. In the second tag, replace the text *your name* with your real name. Make sure your name is enclosed in quotes.
4. Now you need to scroll down to the <description> tag. You need to describe what your Smart Clip does. This description will appear on the Flash Exchange and in the Extension Manager. This section of the file currently looks like this:

   ```
   <description>
   <![CDATA[
   Describe the Smart Clip.
   ]]>
   </description>
   ```

   You want to change the third line to this:

   ```
   <description>
   <![CDATA[
   This Smart Clip allows you to mimic a PowerPoint presentation.
   ➥Every time you press a key on the keyboard, another line of
   ➥your presentation will appear.<br><br>
To use this Smart Clip place an instance of it on the stage, set its
➥options via the Clip Parameters panel.
<br><br>
Configurable options include:
Bullet Type
Bullet Text
Bullet Spacing
   ]]>
   </description>
   ```

Notice that you use the standard HTML
 tag to force line breaks.

5. You also need to let people know where in the Flash user interface (UI) they'll be able to find your Smart Clip after it is installed by the Extension Manager. Smart Clips show up in the Common Library menu after they've been installed. Change the <ui-access> tag to the following:

```
<ui-access>
<![CDATA[
This item is accessed by choosing Window > Common Libraries >
➥SmartBullets.
]]>
</ui-access>
```

6. All that's left to do in the MXI file is to tell the Extension Manager what file you'll be packaging. Change the <files> tag to the following:

```
<files>
    <file source="SmartBullet.fla" destination="$flash/
    ➥Libraries"/>
</files>
```

7. Save your file as **SmartBullet.mxi**.

In the next section, you'll take your Flash file and your MXI file and use the Extension Manager to package them into a format that the Extension Manager can use.

Using the Extension Manager to Package Your File

After you have the file you want to package as an extension and the MXI file created, you can use the Extension Manager to bundle them into an MXP file.

As a general practice, it's a good idea to put the files you are packaging into one folder, just to make sure you have everything together and organized. That way, you don't have to include any path information in your MXI file.

When you're ready, all you have to do is the following:

1. Launch the Extension Manager (Help > Manage Exchange Items).
2. From inside the Extension Manager, choose File > Package Extension.
3. In the Select Extension to Package dialog box, you browse to the MXI file you created (your extension installation file), and then click OK.
4. Next you choose a location for your new MXP and give it a name.
5. Use the Extension Manager to install your new extension.

It's a fairly painless process. Just don't put any spaces in your extension filename and keep the length of the filename to 31 characters or fewer to keep it Macintosh-happy.

You know what to do. Now it's time to do it. In the next exercise, you'll package your extension.

Exercise 22.6 Packaging Your Extension

You'll start by doing a little housekeeping.

1. Create a new folder on your hard drive called **SmartBullet**. Copy the smartBullet.mxi file you just created and the final version of SmartBullet.fla into the new folder. If you were working with the numbered files, rename your finished file **SmartBullet.fla**.
2. From inside of Flash 5, launch the Extension Manager (Help > Manage Exchange Items).
3. Choose File > Package Extension to launch the Select the Extension to Package dialog box.
4. Browse to the SmartBullet folder on your hard drive, select the SmartBullet.mxi file, and then click OK. (See Figure 22.4.)

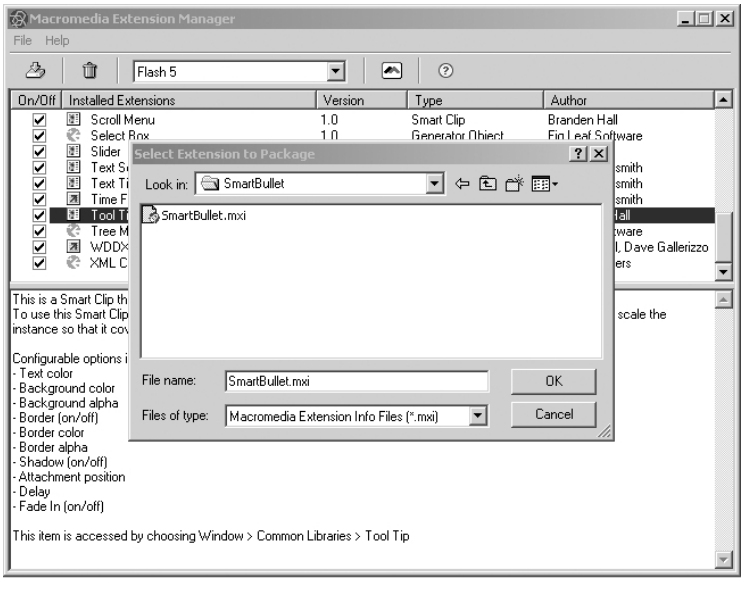


Figure 22.4 In the Select Extension to Package dialog box, you browse to the MXI file you created and then click OK.

5. When the Save Extension Package As dialog box comes up, save your MXP as **SmartBullet.mxp** in the SmartBullet folder.
6. If all went well, you'll get a pop-up message that says "The extension package has been successfully created." Click OK.

That wasn't so bad, was it? But you're not done yet. Before you can send your new extension off to the Flash Exchange, you need to try installing it on your own machine. You also need to test it.

Installing and Testing a New Extension

The first thing you need to do is use the Extension Manager to install your new extension in Flash 5. After that, you need to try the extension out to make sure that it works.

The installation process is easy:

1. Launch the Extension Manager in Flash.
2. Choose File > Manage Exchange Items.
3. Choose File > Install Extension.
4. Browse to the MXP file you just created.
5. Accept the licensing agreement. You should get a pop-up message that says the following: "The 'Smart Bullet Points' extension has successfully been installed."
6. Close the Extension Manager.

With that done, you can test your new extension:

1. Create a new Flash file. You don't have to restart Flash; your menu is automatically updated.
2. Choose Windows > Common Libraries > SmartBullet to open your Smart Clip as a Library.
3. Drag a copy of the SmartBullets Smart Clip onto the Stage.
4. Open the Clip Parameters panel (Window > Panels > Clip Parameters).
5. Make some changes to the values and test your new movie.

You aren't limited just to creating and packaging Smart Clips. You can do the same for several types of extensions.

Installing Other Extensions

The basic process for packaging extensions is pretty much the same for any extension you choose to package. The one item that will be slightly different for each extension type is the <file></file> attribute in the MXI file. There are eight types of extensions you can create. The extensions, general guidelines, and changes to the MXI files are described in the following sections.

ActionScripts

Comment your code! I know hard-core programmers hate to comment code, but tough. Do it anyway. How are other people supposed to understand your brilliant code if you don't tell them what it does?

Please use meaningful variable and function names. Remember that other people are going to be looking at your code. Having incomprehensibly named functions and variables is like having a drawer full of dirty socks. It's bad—on so many levels.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
    <file name="yourFile.as" destination="$flash/ActionScript" />
</files>
```

Next you'll take a look at the Generator object.

Generator Object Extension

Building Generator objects is beyond the scope of this book. For information on guidelines to follow, you can read the "Extending Generator" documentation.

Notice that the Generator object requires three separate files: a definition file, a class file, and a Java file. Also note that the different elements of the Generator object install in different directories for Windows and Macintosh.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
      <file name="yourFile.def" destination="$flash/generator/template"
      ➥"platform="win" />
      <file name="yourFile.class" destination="$generator/classes"
      ➥platform="win" />
      <file name="yourFile.java" destination="$generator/extras/
      ➥YourFolder" platform="win" />
      <file name="yourFile.def" destination="$flash/generator/template"
      ➥platform="mac" />
      <file name="yourFile.class" destination="$flash/generator/classes"
      ➥platform="mac" />
      <file name="yourFile.java" destination="$flash/generator/extras/
      ➥YourFolder" platform="mac" />
</files>
```

You already know how to create custom keyboard shortcut sets in Flash. Did you know that you can package those as extensions as well?

Keyboard Shortcuts

You can create custom keyboard shortcut sets and package them as an extension. Create the shortcut sets in Flash and copy the WFX and MFX files in the Keyboard Shortcuts file to a new folder and then package them.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
     <file name= "yourFile.wfx" destination="$flash/Keyboard Shortcuts"
     ➥platform="win" />
     <file name= "yourFile.mfx" destination="$flash/Keyboard Shortcuts"
     ➥platform="mac" />
</files>
```

You've probably created custom Libraries for use inside Flash. You can package those as well.

Libraries

Any Library files you create should be well organized and use folders. Make sure your Library elements have meaningful and appropriate names.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
    <file name="yourFile.fla" destination="$flash/Libraries" />
</files>
```

The different Publishing Templates for Flash 5 are found on the HTML tab of the Publish Settings dialog box.

Publishing Templates

You also can package custom templates for publishing Flash files.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
    <file name="yourFile.html" destination="$flash/HTML" />
</files>
```

You might find that for some of your extensions, you need to create a sample file to help people understand what to do.

Sample Files

You might decide to create sample files for an extension you are packaging. If you have a single sample file, you can upload it to the Samples folder. Otherwise, you should have the installation file create a new folder for your files.

The changes you need to make to the files attribute in the MXI file are as follows. For the single sample, use this:

```
<files>
    <file name="yourFile.fla" destination="$flash/Samples" />
</files>
```

For multiple samples, use the following:

```
<files>
    <file name="yourFirstSample.fla" destination="$flash/
    ➥YourSampleFolder" />
    <file name="yourSecondSample.fla" destination="$flash/
    ➥YourSampleFolder" />
    <file name= "yourThirdSample.fla" destination="$flash/
    ➥YourSampleFolder" />
</files>
```

You've already taken a close look at how you package a Smart Clip.

Smart Clips

Make sure you put all the assets for your Smart Clip in a single asset folder. If you've created a custom user interface for the Parameters panel, you also need to include the user interface FLA and SWF files. Otherwise, you can omit those two lines.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
    <file name= "yourFile.fla" destination= "$flash/Libraries" />
    <file name= "yourUI.fla" destination= "$flash/Libraries/Smart Clip
    ➥UI" />
    <file name= "yourUI.swf" destination= "$flash/Libraries/Smart Clip
    ➥UI" />
</files>
```

Finally, you also can submit projector files as extensions.

Utility Files

If you decide to submit a Flash projector as a utility, you have to package the original FLA in the MXP so that it can be evaluated by Macromedia.

The changes you need to make to the files attribute in the MXI file are as follows:

```
<files>
    <file name="yourFile.fla" destination="$flash/YourFolder" />
    <file name="yourFile.fla" destination="$flash/YourFolder" />
</files>
```

The MXI file you create might contain more than one type of extension. For example, if you wanted to package a Smart Clip and several samples of how to use the Smart Clip, the files attribute in the MXI file would look like this:

```
<files>
    <!—Smart Clip Files ->
    <file name= "yourFile.fla" destination= "$flash/Libraries" />
    <file name= "yourUI.fla" destination= "$flash/Libraries/Smart Clip
    ➥UI" />
    <file name= "yourUI.swf" destination= "$flash/Libraries/Smart Clip
    ➥UI" />
    <!—Sample Files ->
    <file name="yourFirstSample.fla" destination="$flash/
    ➥YourSampleFolder" />
    <file name="yourSecondSample.fla" destination="$flash/
    ➥YourSampleFolder" />
    <file name= "yourThirdSample.fla" destination="$flash/
    ➥YourSampleFolder" />
</files>
```

That sums up most of what you need to know about packaging extensions for distribution. If you really want to get into the nitty-gritty details, download the MXI File Format PDF on Macromedia's Web site.

Summary

As you've seen, creating Smart Clips and packaging them for distribution, either on the Flash Exchange or just between developers, is not a difficult task. To get more ideas of the types of Smart Clips that other developers are creating, make sure you visit the Flash Exchange at `http://www.macromedia.com/exchange/flash`.