

11

Audio Compression with Shockwave



Free audio Xtras from Macromedia

The Shockwave Audio Xtra WAV file converter for Director for Windows 95/NT and the SoundEdit Streaming Audio Toolkit for Power Macintosh are available from the Macromedia Web site at: <http://www.macromedia.com/shockwave/devtools.html>.

Note: The Shockwave Audio compression Xtras for internal Director sounds are included in each of the Afterburner for Director packages.

The first time I saw Shockwave I was immediately struck, not by the fact that an interactive Director movie was playing on a Web page, but that there was music and sound playing on the page.

Back then, sound on the Web was always very intrusive — not at all a seamless part of the user experience. You had to click on a link to a sound file, which would then download; a helper application would then open — taking you away from the Web page — and play the sound. Of course there are now dozens of ways to add sound seamlessly to a Web page — but at the time of Shockwave's release the fact that you could get sound so transparently through a Web page reinforced to me just how effective sound can be in enhancing the experience of the user.

Of course all this wonderful sound came at a high cost: Shockwave's AfterBurner offered no compression for audio. Thus adding an audio file to a Shockwave movie increased the size of the movie by the file size of the audio. And, as you may know, even the simplest of sounds can produce very large file sizes. That all changed in July 1996 with the release of Shockwave Audio.

Now, not only can internal Director sounds be compressed as much as 176:1 using the Shockwave Audio compression Xtra, but audio files can be streamed over the Internet.

Compressing sounds with Afterburner

With the Shockwave Audio Xtras available free from Macromedia's Web site, you can compress sounds imported into the cast of a Director movie.

Shockwave sells tools

When Shockwave Audio was introduced, Windows users were dismayed to learn that there was no way to create streaming Shockwave Audio files on a PC. The popular consensus among Mac users was that it was Macromedia's way of making up to them for their lack of a Mac plug-in at the initial release of Shockwave for Director in 1995. Actually, the reason was much simpler than that. Marcos Sanchez, former product manager for Shockwave put it plainly, "Shockwave drives tool sales." Macromedia has no sound tool for Windows, hence no streaming audio development on Windows. Soon after the initial release of Shockwave Audio, Macromedia released an Xtra for Director for Windows that allows users to create streaming audio files from within the application. Sounds like it would be a great thing for Mac users as well, but don't hold your breath; Macromedia still sells Sound Edit 16, a sound editing application.

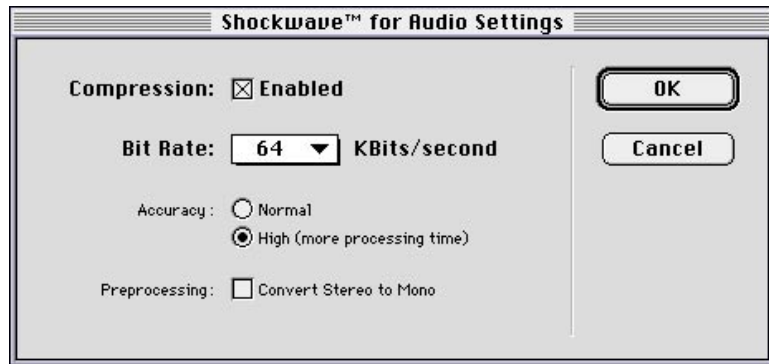


Figure 11-1. The *Shockwave for Audio Settings* dialog box.

To enable audio compression in Director, select *Xtras/Shockwave for Audio Settings* in Director (Figure 11-1). Enable audio compression by selecting the check box. Next select a bit rate from the popup menu. The lower the number, the better the compression and poorer the sound quality. Next, select an accuracy setting. Choosing *Convert Stereo to Mono* will further reduce file size and quality.

Although Shockwave compression is very good, using internal sound should be reserved for small sound clips. This is because

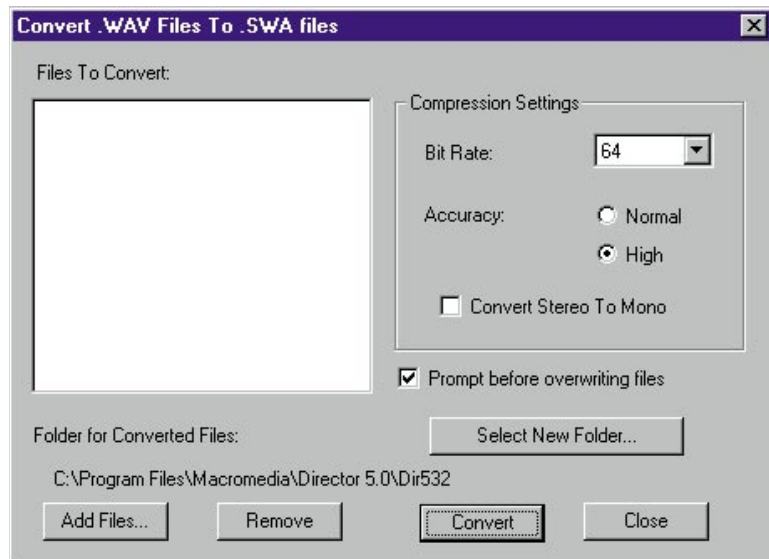


Figure 11-2. The *Convert .WAV Files To .SWA Files* dialog box.

internal sounds are embedded within the shocked movie, which must be fully downloaded to begin playing. Large audio files, such as songs or long narratives, can be compressed and streamed over the Internet using your Director movie as a player.

Creating streaming audio files in Windows

Windows users can convert WAV files to streaming audio files (SWA) right in Director. Select **Xtras/Convert WAV to SWA**. This brings up the dialog box shown in Figure 11-2. Next, click on the *Add Files* button to bring up an *Open File* dialog box, where you can select the files you want to convert. Then select a bit rate and accuracy setting and click *Convert*.

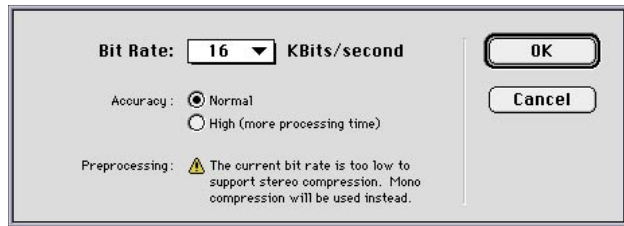


Figure 11-3. The *Shockwave for Audio Settings* dialog box in Sound Edit 16, seen here, is nearly identical to the one in Director (Figure 11-1).

Creating streaming audio files on a Macintosh

To create streaming audio files on a Macintosh, you must use Sound Edit 16 from Macromedia. To create an SWA file in Sound Edit, open your sound clip and select **Xtras/Shockwave for Audio Settings**. This brings up a dialog box nearly identical to the one in Director (Figure 11-3). Select the desired bit rate and accuracy settings and click *OK*. Next, select **File/Export**, which brings up the *Export* dialog box seen in Figure 11-4. Name your file and select *SWA File* from the *Export Type* popup menu.

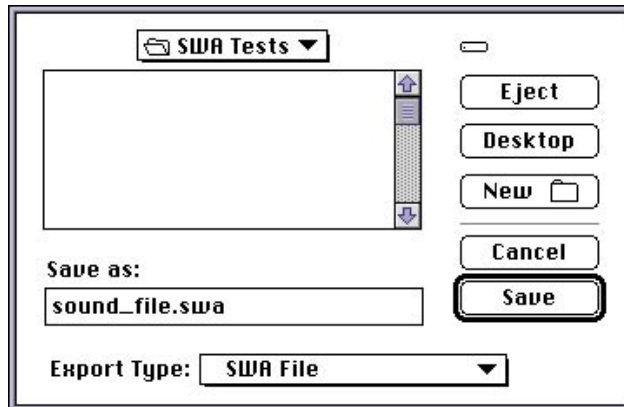


Figure 11-4. Sound Edit 16's *Export* dialog box.

Once you've created your streaming audio files, you can upload them to your Web server and play them from your Shockwave movie. Let's take a look at our next example to see how this is done.

Listen, it's OddoRadio

Tommy Oddo, a Texas-based Web designer, created an interesting audio player in Director as part of a self-promotional Shockwave movie called *OddoRadio*. When you first arrive at *OddoRadio*,

Oddo Design

<http://www.oddo.com/>

Tommy Oddo began his career as a graphic designer over 17 years ago. After attending the Art Institute of Atlanta, he relocated to Houston where he runs his Web design and marketing business. His work has garnered many design and illustration awards and has been featured in several books. You can find out more about Tommy by tuning in to his Web site.

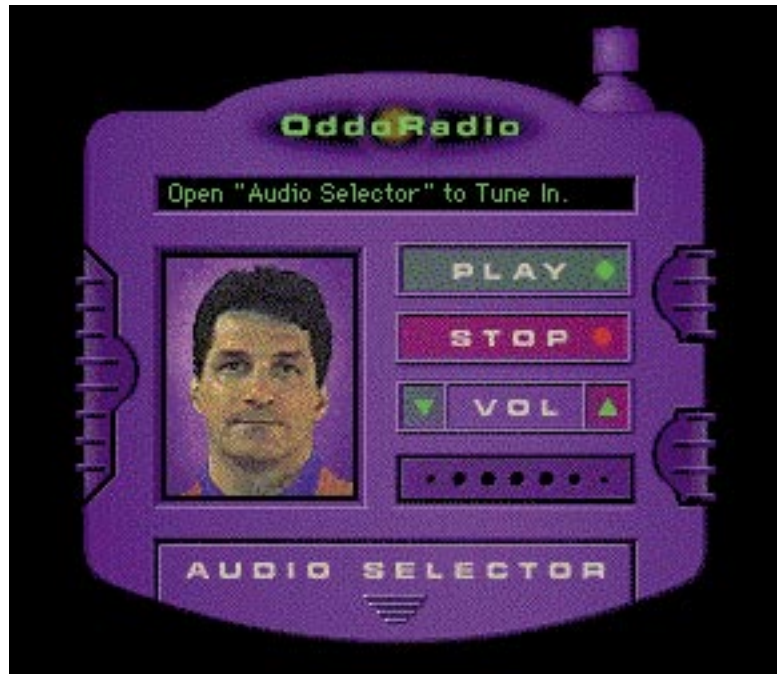


Figure 11-5. The *OddoRadio* interface.



Figure 11-6. *OddoRadio* with the *Audio Selector* drawer open.

you're presented with a fun, techy looking interface reminiscent of something you might get as a Happy Meal toy from McDonald's (Figure 11-5). Click on the *Audio Selector*, and a drawer full of audio selections pops open (Figure 11-6). Clicking on one of the selections causes the drawer to snap shut and the first portion of the audio to preload as a buffer. Once the buffer is preloaded, you can click the *Play* button to start the audio stream.

Let's take a look inside *OddoRadio* to see how streaming audio works.

First, in order to use streaming audio files with your Director movie, an *SWA file Xtra cast member reference* must be placed in the cast of the movie. Select **Insert/Other/SWA Streaming Xtra**. This reference file appears as cast member 4 in the *OddoRadio* movie (Figure 11-7) and will be used later to specify the URL of the streaming audio file.

When the user makes a selection from the *Audio Selector* drawer, it calls a *mouseUp* script attached to the graphic. This script simply calls a custom handler for the particular audio

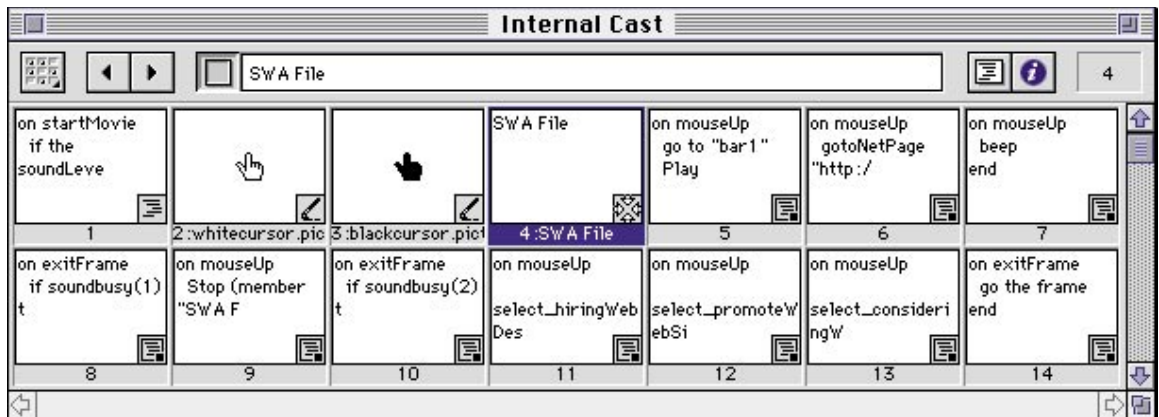


Figure 11-7. The SWA file Xtra cast member reference appears as cast member 4 in *OddoRadio's* cast window.

selection. In this example, we'll look at the script for the *About OddoRadio* button, which looks like this:

```
on mouseUp
    select_aboutOdRadio
end
```

This script has just one purpose: to call the *select_about OdRadio* handler seen in Figure 11-8.

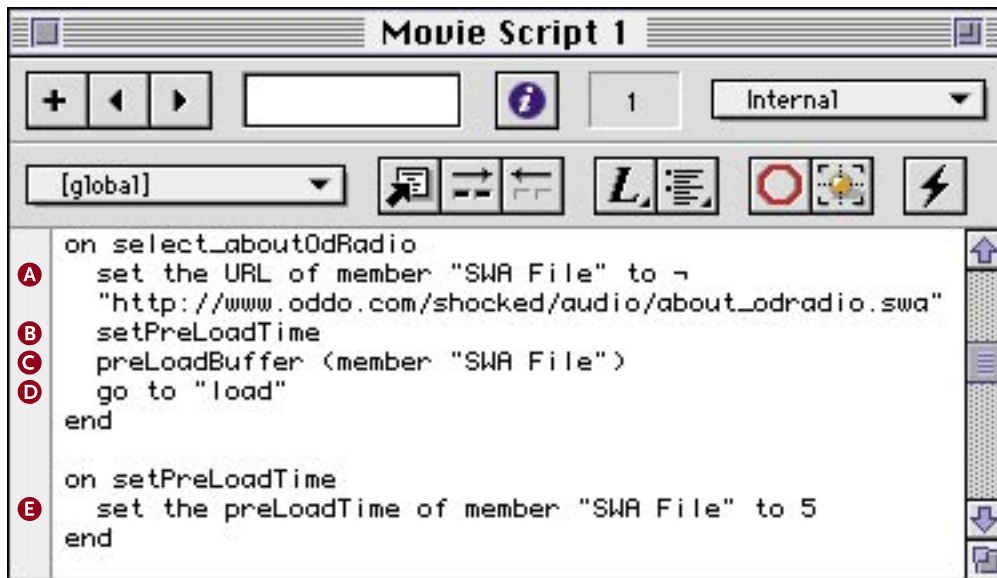


Figure 11-8. The *select_aboutOdRadio* handler.

The first line of the `select_aboutOdRadio` handler sets the URL of the cast member called *SWA File* to the URL of the `about_odradio.swa` file. **A** This cast member is the SWA file Xtra reference that was inserted into the cast earlier. Next, the script calls another handler **B** named `setPreLoadTime`.

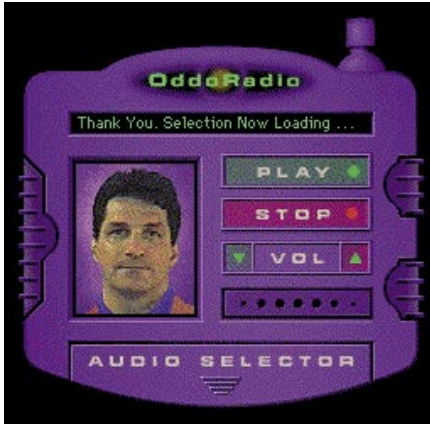


Figure 11-9. The *OddoRadio* status window during the load frames.

This handler, also shown in Figure 11-8, **E** simply sets the preload time of the SWA file to 5 seconds. That is, the file will download for 5 seconds before starting to play. After setting the preload time, the handler ends and the `select_aboutOdRadio` handler resumes. The next line in this handler **C** uses the `preLoadBuffer` command to start the download of the specified portion of the SWA file into memory. Lastly, the handler sends the movie to the frame called `load`. **D**

This frame starts a four-frame series that animates the ellipsis at the end of the words “Thank You. Selection Now Loading...” in *OddoRadio*'s status window (Figure 11-9). This gives

OddoRadio compression

Shockwave Audio compression can produce some fairly astounding results, as evidenced by the files in Tommy's OddoRadio. Tommy recorded all his sounds at 16 bits/44.100 kHz and downsampled them to 16/22.050. After equalizing and normalizing the tracks in Sound Edit 16, Tommy exported the sounds as SWA files at a bit rate of 16 K/Bits per second achieving the amazing results below:

File name	Original size	SWA size
about_odradio.swa	3.0MB	141K
hire_firm.swa	7.0MB	326K
promote_web.swa	4.7MB	223K
want_site.swa	4.2MB	197K
web_cost.swa	4.8MB	227K
where_begin.swa	4.2MB	197K

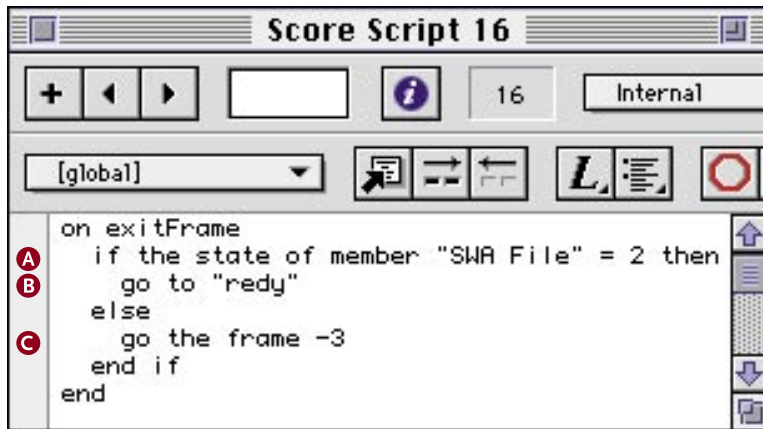


Figure 11-10. The *load frames'* *exitFrame* handler.

the user a clue that something is happening in the background, and that the movie isn't simply frozen. The last of the frames in this series calls the *exitFrame* handler seen in Figure 11-10 when the movie exits the frame.

This script uses an *if...then* statement **A** to check if the state of the SWA file is 2. The Lingo *the state of member* checks the "state" of an SWA file and returns a numerical code depending on the state. The number 2 means that preloading is finished. (For a list of possible states and their numeric codes, see Appendix B.) If the preload is done, the movie goes to the frame called *redy*. **B** If preloading is not done, the movie jumps back 3 frames **C** — back to the start of the ellipsis animation.

Once preloading is finished and the movie has moved to the *redy* frame, it now waits for the user to press the *Play* button, while the words "Selection Ready. Please Press Play" blink in the status window (Figure 11-11). Tommy adds another nice visual touch at this point by raising the antenna on the radio.

When the user presses *Play*, the script in Figure 11-12 is activated. This script sends the movie to the frame called *bar1* **A** and plays the SWA file. **B**

The *bar1* frame starts a series of frames that, like the *load* frames, animate the ellipsis at the end of the words "Thank You."

Why buffer?

When audio is streamed over the Internet, it is necessary to send a certain amount of audio data to the user's computer memory before play begins. This preloaded data is called a buffer. In order to ensure an uninterrupted stream of audio, the player plays the data from this buffered memory, allowing more data to download while it is being played. The higher the buffer is set, the better chance you will have of uninterrupted audio, however, the higher the buffer, the longer the user will have to wait before playback begins.

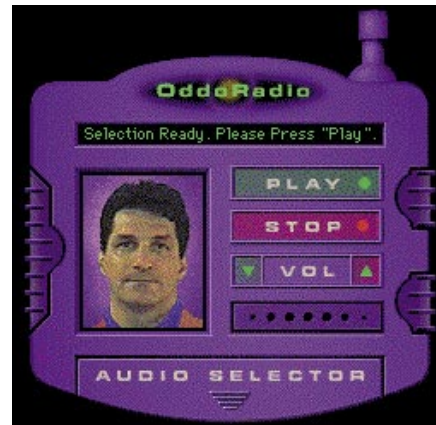


Figure 11-11. Waiting for the user to press "Play." Note the antenna is now raised.

See it in action

OddoRadio is located on the CD in *source/odradio.dir*.



Figure 11-12. The *Play* button script.

Selection Now Playing...." As the movie passes through each of the frames in this series, the *exitFrame* handler shown in Figure 11-13 is called. This handler checks to see if the SWA file has finished playing. An *if...then* statement **A** checks the state of the SWA file. A state of 5 means the file is done playing. Then a *Stop* command is issued **B** and the movie goes to the frame called *strtc*, the *Audio Selector* frame. If the file is not done playing, the movie jumps back to the beginning of the ellipsis animation. **C**

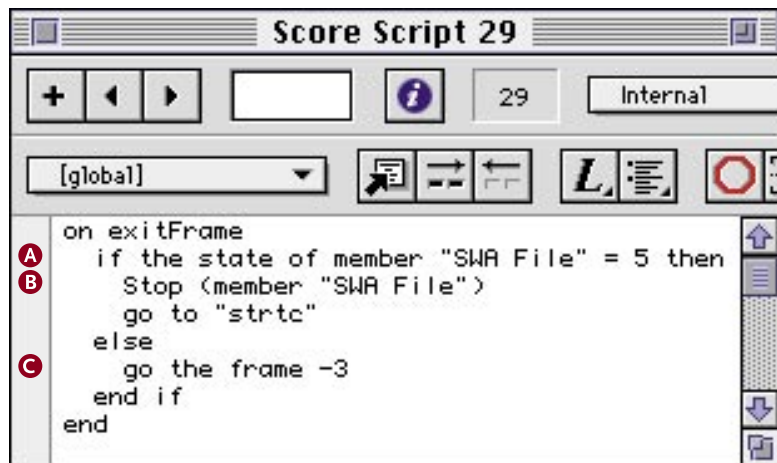


Figure 11-13. The *bar 1* frames' *exitFrame* handler.



Figure 11-14. Raspberry Media's Shockwave Audio player.

Using external parameters

In the example above, the URLs for the SWA files are hard-coded in the Shockwave file. While this is a perfectly fine method, it has several drawbacks. If you want to change the selections in the file, or if you move the SWA files to another location on your server, the original movie must be edited, reburned, and uploaded. Also, if you wanted to use the same player on different pages, with different audio selections, a new movie for each page would have to be created by you, and downloaded by the user.

Using Lingo, Shockwave movies can access parameters that are included in the *EMBED* or *OBJECT* tags of an HTML document. By changing these external parameters, you can easily alter the functionality of your movies, including URL links and preload times for SWA files, as we'll see in our next example.

Raspberry Media has created a Shockwave audio player to showcase some of the examples of their professional sound design services (Figure 11-14). The Lingo for the Raspberry player, written by Josh Kopel of Articus Media Labs, uses external parameters to transfer the base URL for a group of SWA files, the list of SWA file names, the preload time for the files, and a list of text names for each of the songs into the Shockwave movie.

The player's external parameters are located in the *EMBED* tag of the HTML page (Figure 11-15). The parameters are read

See it in action

The Raspberry Shockwave player is located on the CD in *source/raspberry.dir*.



Raspberry Media

<http://www.raspberrymedia.com/>

Based in Sebastopol, CA, Raspberry Media is a design studio specializing in Web site development, graphic design, digital audio, and photography.

Over the past few years they have completed projects for an international clientele including Virgin Interactive, *Rolling Stone*, GQ, EMI Records, Mindscape, and the University of California.



Articus Media Labs

<http://www.dev-com.com/~amlab/>

Joshua A. Kopel is one of the directors of Articus Media Labs, Inc. a Web and multimedia development firm located in Philadelphia, PA.

```
<HTML>
<HEAD>
<TITLE>Sound Effects</TITLE>
</HEAD>
<BODY BGCOLOR="#000000">

<EMBED width=416 height=171 SRC="rasp_player.dcr"
swURL="http://www.raspberrymedia.com/audio/"
swList="sound_eff.swa,blues.swa,reggae.swa"
swTEXT="Sound effects Medley,Blues Medley,Reggae Medley"
swPreLoadTime=4
animate = "YES">

</BODY>
</HTML>
```

Figure 11-15. The HTML for Raspberry Media's player contains several external parameters in the *EMBED* tag.

into the Shockwave movie using the *if...then* statements shown in Figure 11-16. The first of these statements **A** checks to see if there is an external parameter named *swText* in the *EMBED* tag, using the Lingo command *externalParamName*. If there is a parameter named *swText*, the string of text contained between the quotes in the *EMBED* tag is put into a variable named *myNameList* using the Lingo *externalParamValue*. **B** If there isn't an *swText* parameter in the *EMBED* tag, a handler called *doerror* is executed **C** and the process of reading in the parameters is aborted. **D**

The *doerror* handler

The *doerror* handler (Figure 11-17), uses a *case* statement to display a dialog box to alert the user of the error. When the *doerror* handler is called from the *if...then* statement, a value of 1 is passed along to the *doerror* script. This value will tell the *doerror* script which error to display.

In the first line of the *doerror* script a variable called *what* is created to capture the value being passed from the *if...then* statement. **A** This variable is then used in the *case* statement **B** to determine which of the instances in the *case* statement to execute. In this instance, *what* has a value of 1, so the first line in the *case* statement is executed. **C** This line **D** brings up an alert box with the sentence, "Cannot find the list of song names." The

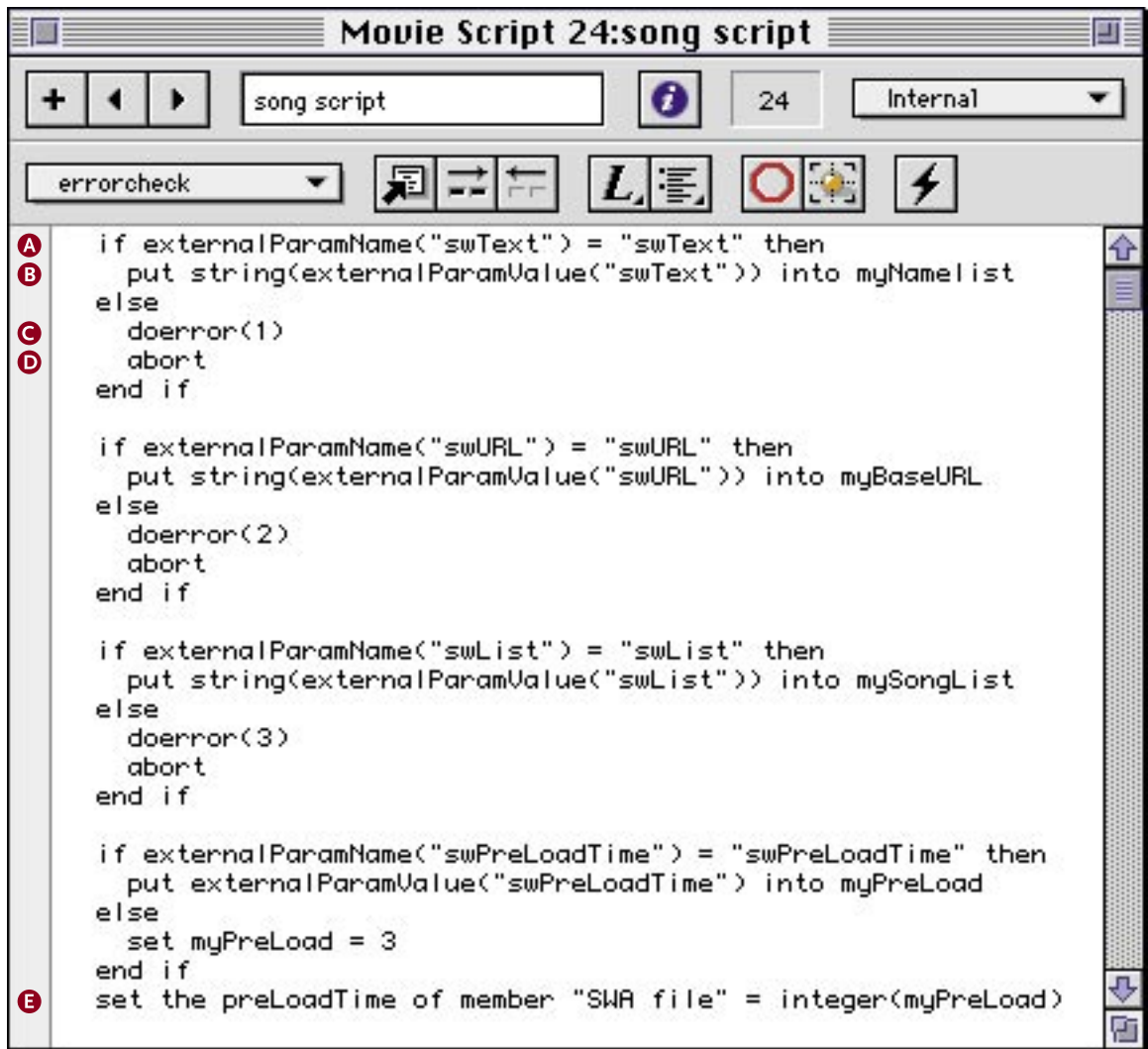


Figure 11-16. The Raspberry Media player uses *if...then* statements to capture the external parameters into variables.

doerror script then aborts. **E** The *abort* command causes the script to exit the *doerror* handler, without executing any further commands.

Capturing the remaining parameters

If there is no error in the first *if...then* statement of Figure 11-16, the rest of the statements are executed, capturing the remaining parameter values from the *EMBED* tag into separate variables.

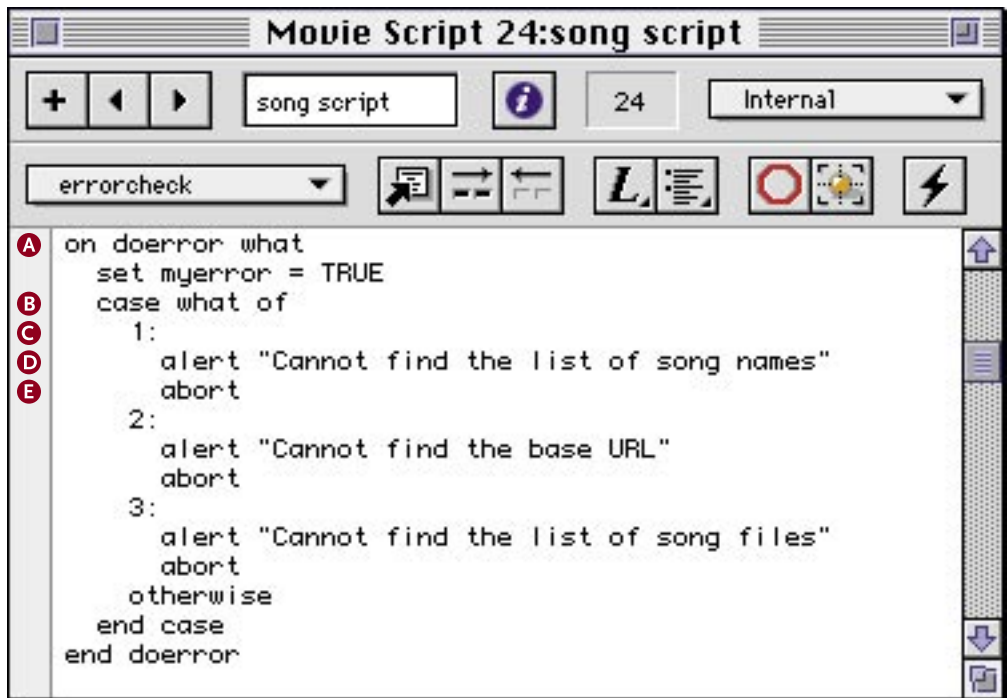


Figure 11-17. The *doerror* handler.

Once the parameters have been captured into the variables, they can be used to set preload times and the URLs of the songs, much as they were in *OddoRadio*. An example of this is seen in the final line of code shown in Figure 11-16. This line **E** sets the *preloadTime* of the SWA file to the integer contained in the variable called *myPreLoad*.

This is an excellent example of how you can customize Shockwave movies simply by altering external parameters in the HTML document.

Before Shockwave audio compression became available, designers were justified in ignoring this very important aspect of the overall user experience. Now, with the amazing amount of compression obtainable with Shockwave Audio, as well as the ability to stream, it is no longer necessary to neglect sound as you plan your next Shockwave project.