MX

macromedia®
FLASH™ MX Professional
2004

Flash Lite Authoring Guidelines for the i-mode Service by NTT DoCoMo

Acknowledgments

Director: Erick Vera

Producer: Barbara Nelson

Writing: Paul Goldman

Editing: Lisa Stanziano

Print Design and Production: Adam Barnett

First Edition: March 2003

Macromedia, Inc.
600 Townsend St.
San Francisco, CA 94103

CONTENTS

CHAPTER 1
Introduction

Macromedia has created a new profile of the Flash Player called Macromedia® Flash™ Lite, designed for consumer mobile devices, including phones for the i-mode service by NTT DoCoMo. This format is designed to run optimally on devices with limited memory, processor speed, and display area. Content created for Flash Lite is most similar to Flash Player 4 content.

The *Macromedia Flash MX Professional 2004 User Guide for Flash Lite* describes, in general, tools and guidelines for authors creating Flash Lite movies. This document contains authoring information specific to creating movies for i-mode phones.

About the i-mode service by NTT DoCoMo

The i-mode service by NTT DoCoMo is a mobile phone service in Japan that provides its customers with both voice and comprehensive data services. With an i-mode phone, users can exchange i-mode e-mail and obtain information from i-mode menu sites and i-mode compatible Internet sites.

The i-mode phone contains a browser that displays i-mode compatible HTML web pages. Beginning with the 505i phones, you can view Macromedia Flash Lite movies from the i-mode browser. You can also use the phones' My Picture and Standby Screen applications to view Flash Lite movies. Although a number of manufacturers produce 505i phones, all of them support the same Flash Lite functionality.

Getting started

To create Flash Lite movies for i-mode phones, you need to install Macromedia Flash MX Professional 2004, available from the Macromedia website. (See Appendix D, "References," on page 47, for links to the Macromedia website.) . Read the *Macromedia Flash MX Professional 2004 User Guide for Flash Lite* for an overview of Flash Lite and basic authoring information.

To test your completed Flash Lite movies for i-mode phones, you should obtain the i-mode HTML Simulator from the DoCoMo website. (See Appendix D, "References," on page 47.) The Simulator is an application tool that emulates the operation of an i-mode phone and allows you to test the validity of Flash Lite movies. Though useful, the Simulator is no substitute for testing on actual i-mode phones—only testing on actual phones will give you a true picture of your Flash Lite movie's performance.

CHAPTER 2
Developing Content

Starting with the 505i phones, the i-mode service by NTT DoCoMo supports the ability to view Flash Lite movies. The same Flash Lite functionality is available on all 505i phones, regardless of manufacturer. This chapter describes considerations for creating Flash Lite movies that run on i-mode phones, from general functionality to performance and size constraints.

The 505i phones support Flash Lite in both English and Japanese. However, there are a few exceptions to the standard Flash Lite specification. To review the standard specification, see the *Macromedia Flash MX Professional 2004 User Guide for Flash Lite*. The exceptions to the standard are detailed in this document.

Navigation and key events

Flash Lite for i-mode uses three keys for navigation: Up, Down, and Select. The Left and Right keys are reserved for the i-mode browser. These three keys correspond to the Shift+Tab, Tab, and Enter keys on the desktop versions of the Flash Player.

The keys 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, and # are also available. These correspond to the same keys on the desktop versions of the Flash Player. You can attach ActionScript to these keys and the Enter key as you would normally in Flash. ActionScript attached to other keys is ignored.

Text and fonts

Flash Lite includes support for both device and embedded fonts. You can use embedded fonts to give you more control over the design of your movie, but doing so increases the SWF file size. Using the device font for text limits you to a single font, but helps keep your file size small.

When using device fonts, Flash Lite limits special text formatting for dynamic text fields to justification (left, center, right) and color. Formatting options such as superscript, subscript, kerning, bold, and italic are not supported.

Flash Lite does not support input text fields—input text fields are not selectable and cannot be used to enter text.

Emoji

i-mode phones support special pictographic characters called Emoji. The following characters are examples of Emoji:

These are encoded by replacing characters in the standard Shift-JIS table. As long as the phone's font supports Emoji encoded as standard Shift-JIS characters, Flash Lite can display them.

When authoring using Flash, you first need to ensure that you have Shift-JIS fonts installed on your desktop authoring system. You also need to select a Shift-JIS font for the text fields in which you want to display Emoji characters.

Note: The phone's font set controls the color and appearance of Emoji characters.

For further information about Emoji, see Appendix D, "References," on page 47.

i-mode compatible HTML

i-mode browsers can directly run Flash Lite movies, or movies can be embedded in i-mode compatible HTML web pages.

The i-mode compatible HTML specification is based on a subset of HTML 2.0, HTML 3.2, and HTML 4.0 specifications that DoCoMo extended with tags and attributes for special use on mobile phones. As an example, extensions include the `tel` URL protocol, which is used to link to a phone number and let users initiate a phone call.

For information on i-mode compatible HTML, see Appendix D, "References," on page 47.

ActionScript and properties

Flash Lite for i-mode supports most Flash 4 ActionScript commands. The following are notable exceptions:

- Use the `add` operator instead of the `&` command to concatenate strings.
- Button mouse events such as `dragOver`, `dragOut`, and `releaseOutside` cannot be used to trigger ActionScript attached to buttons. However, in addition to keypress events, the events `press`, `release`, `rollOver`, and `rollOut` can be used to trigger ActionScript when attached to buttons and accessed through key-based navigation.
- Draggable movie clip functions and properties (for example: `startDrag`, `stopDrag`, and `_dropTarget` properties) are not supported.
- Use the `eq` operator to compare strings and the == operator for numeric comparison.
- URL encoding must be done manually using ActionScript. The `escape()` ActionScript function is not a Flash 4 function and is not available in Flash Lite.
- The `fscommand()` function is not supported.
- The default Quality level for Flash Lite during playback is medium and there is no support for bitmap smoothing.

- The `getURL()` function can only be called once per keypress, and can only be used for the `http`, `mailto`, `https` and `tel` protocols. Only the first `getURL()` call in a keypress statement block is executed; all subsequent `getURL()` calls in the same block are ignored.
- A button action can be assigned to launch an e-mail composition window with the address, subject, and body text fields already populated. There are two methods to do this. Method 1 can be used for either Shift-JIS or English character encoding, while method 2 only supports English character encoding.

  Method 1

  Set variables for each of the desired parameters, for example:

  ```
  on (release, keyPress "#") {
    subject = "email subject";
    body = "email body";
    getURL("mailto:somebody@anywhere.com", "", "GET");
  }
  ```

  Method 2

  Define each parameter within the getURL action, for example:

  ```
  on (release, keyPress "#"){
    getURL("mailto:somebody@anywhere.com?subject=email subject&body=email
    body");
  }
  ```

- Key events can only be attached to the keys 0-9, #, *, and the Enter key.
- The `loadMovie()`, `loadVariables()`, `loadMovieNum()`, and `loadVariablesNum()` functions are not supported.
- The `MaxScroll` and `Scroll` text-scrolling properties are not supported.
- Sound functionality is limited to event sound. An event sound can only be triggered to play when it is attached to a keypress event. Only the first event sound in a keypress statement block is played, and all other subsequent sounds in the same block are ignored.
- There is no synchronized audio, so the `_soundBuffTime` property is not supported.
- The range of valid integer numbers that can be represented is -2,147,483,648 to 2,147,483,647.
- Math functions are not natively supported. In Flash Lite, the methods and properties of the Math object are emulated using approximations and may not be as accurate as the non-emulated math functions supported in Flash Player 5 and above.
- The following Math functions can only be used with constants, not variables: `Math.acos()`, `Math.asin()`, `Math.atan()`, `Math.atan2()`, `Math.cos()`, `Math.pow()`, and `Math.tan()`.
- The `_url` property is not supported.
- The `Number()` and `String()` functions are not supported.

  Note: Flash 4 ActionScript does not support arrays. However, they can be emulated using the eval() function. For more information, see Macromedia TechNote 14219, "How to use Eval to emulate an array," at www.macromedia.com/go/flash_support (English) or www.macromedia.com/go/flash_support_jp (Japanese).

ActionScript commands that are not recognized are ignored. For a detailed listing of supported ActionScript and properties, see Appendix A: "Supported ActionScript" on page 25 and Appendix B: "Supported Properties" on page 37.

Sound

Flash Lite for 505i phones does not support the standard Flash Player audio formats—Raw, ADPCM or MP3. Instead, only MFi (Melody Format for i-mode) is supported. In addition, each manufacturer's 505i phone supports the standard MFi format, plus its own proprietary extensions.

Flash Lite does not support streaming sound, sound mixing, or looping of sound. Only event sound is supported and only one sound can be played at a time.

For detailed information about embedding sound into Flash Lite movies for i-mode phones, see Chapter 3, "Sound," on page 15.

Network access

The Flash Lite specification for i-mode supports the `getURL()` function in a restricted manner. The `getURL()` function is ignored unless the user first presses one of the following keys: 0-9, *, #, or the Select key. Only the first `getURL()` call in a keypress statement block is executed; all subsequent `getURL()` calls in the same block are ignored.

The `getURL()` function can be used to load another SWF or HTML page (`http`), a secured (SSL-Secure Sockets Layer) HTTP page (`https`), send e-mail (`mailto`), or dial a phone number (`tel`).

Screen size

The i-mode phone screen size is one of the most important factors to keep in mind when developing Flash Lite movies for i-mode phones. Generally, content looks better scaling up, rather than scaling down, so it is best to create content for the smallest screen area. The screen area available to Flash Lite varies from phone model to phone model, and across the applications featuring Flash Lite. In order for a Flash Lite movie to look the best in the browser on all 505i phones, a resolution of 240x240 is recommended. The screen area available to Flash Lite in the My Picture and Standby Screen applications varies depending on the specific 505i phone.

Detailed information on the screen area available to Flash Lite on i-mode phones is available on the DoCoMo website. (See Appendix D, "References," on page 47).

Movie size

There are limitations on file size and run-time memory usage for Flash Lite movies running on i-mode phones. There is a prescribed limit on how large a web page can be, whether it includes Flash Lite movies or not. For 505i phones, this limit is 20KB. Full details can be found at the DoCoMo website (see Appendix D, "References," on page 47). This limit applies to an i-mode page's HTML, SWF content, and all graphic images combined. Web pages larger than this limit cannot be downloaded to an i-mode phone and no error message appears. This limitation also applies to Flash Lite movies played directly in the browser without being embedded in an i-mode compatible HTML file.

The run-time memory available to Flash Lite movies running on i-mode phones is limited and may vary from model to model. Generally, for the 505i phones, this limit is not less than 200KB. Because Flash MX Professional 2004 does not provide a mechanism for checking a phone's run-time memory consumption, Macromedia strongly recommends that you test all content on actual i-mode phones.

Performance Optimization

CPU speed in i-mode phones varies from model to model, and is typically much slower than current desktop computers. Therefore, it is extremely important to consider movie performance and optimization from the beginning of each project. The optimization recommendations for creating any Flash movie also apply to Flash Lite movies created for i-mode phones. For the latter, their importance is amplified.

Note: In Flash MX Professional 2004, you can find tips on optimizing Flash movies—select Help > Using Flash -> Search and enter **optimizing movies** in the keyword search text box.

If you follow some simple guidelines, as described in this document, to author your movies, you can create rich and compelling content despite CPU limitations.

Sound

Since Flash MX Professional 2004 does not natively support MFi, you must temporarily substitute a proxy sound in a recognized format such as MP3. Details and procedures on sound substitution for i-mode phones and Flash Lite are presented in Chapter 3, “Sound,” on page 15.

Animation

When creating animated content for an i-mode phone, it is important to keep in mind the phone’s CPU limitations. The following guidelines can help prevent your movie from running slowly:

- If you need to provide intense or complex animation, experiment with changing the quality setting of the movie. The default quality setting is Medium.

  To change the quality setting in Flash MX Professional 2004, select File > Publish Settings, then select the HTML tab. Select a quality setting from the Quality pop-up menu.

  Because changing the quality setting may noticeably affect the visual quality of the movie, be sure to thoroughly test the movie.
- Limit the number of simultaneous tweens.
- Alpha effects on symbols are very CPU intensive and should be used sparingly. In particular, it is generally not a good idea to tween symbols that have alpha levels that are not fully opaque (less than 100%).
- Avoid intensive visual effects. These include large masks, extensive motion, alpha blending, extensive gradients, and complex vectors.
- Although animating with ActionScript may produce more desirable results, in general, you should avoid unnecessary use of ActionScript.
- Experiment with combinations of tweens, key frame animations, and ActionScript-driven movement to produce the most efficient results.
- Test animations frequently on your target phones whenever possible.

Using bitmaps

Although some i-mode phones may have more than 16 bits of color resolution, Macromedia recommends optimizing bitmaps to 16 bits before importing them into Flash MX Professional 2004. Doing so reduces Flash Lite movie size and gives you more control over the final output. Also, make sure that bitmaps are imported at the size they need to be in the Flash Lite movie. Using larger than required bitmaps results in higher run-time memory requirements.

Bitmaps versus vectors

Flash Lite generally uses vectors to define content, which can tax a phone's CPU when rendering complex graphics and animations. In general, the more vectors that are manipulated on the stage, the more CPU power is required. This is also true for Flash movies delivered on desktop machines. However, i-mode phones are far less powerful than desktop machines and more care should be taken to avoid taxing the CPU.

When creating content for i-mode phones, it is sometimes better to use bitmaps instead of vectors because they require less CPU power to animate. For example, a road map of a large city would have too many complex shapes to scroll and animate well on an i-mode phone if it were created as a vector graphic; a bitmap would work much better.

Using bitmaps produces larger files, so take care during development to find the right balance of CPU versus file size and run-time memory requirements. Because of mobile phones' smaller screens, slower data transmission speeds, limited memory and CPU speeds, developers should take extra care in planning and testing.

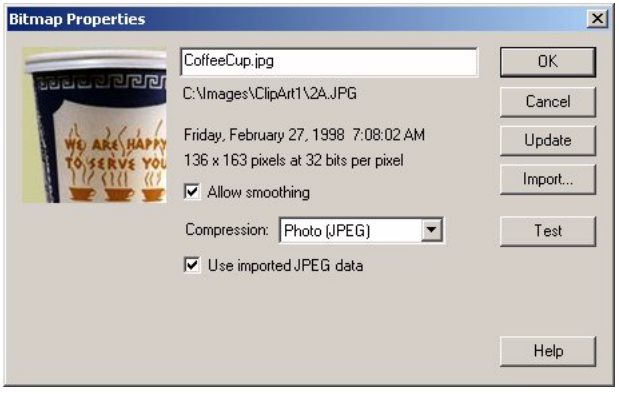
If you are using bitmaps, you can set image compression options that will reduce your SWF file size.

To set bitmap image compression:

1 Select a bitmap in the Library window.

2 Right-click (Windows) or Control-click (Macintosh) the bitmap's icon in the Library window.

3 Choose Properties from the options menu. The Bitmap Properties dialog box appears:



- Select Photo (JPEG) in the Compression pop-up menu for images with complex color or tonal variations, such as photographs or images with gradient fills. This option produces a JPEG format file.

Select the Use Imported JPEG Data checkbox to use the default compression quality specified for the imported image. To specify a new quality compression setting, deselect Use Imported JPEG Data and enter a value between 1 and 100 in the Quality text box. A higher setting produces a higher image quality, but also a larger file size, so adjust the value accordingly.

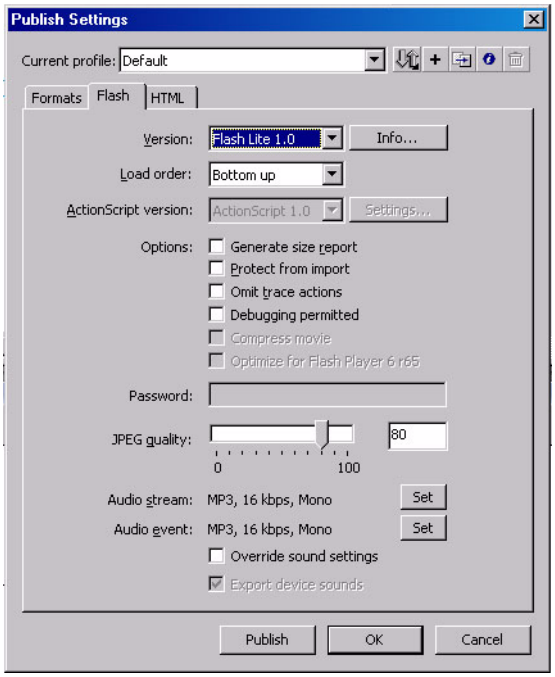
- Select Lossless (PNG/GIF) in the Compression pop-up menu to compress the image with lossless compression, in which no data is discarded from the image. Use lossless compression for images with simple shapes and relatively few colors. Save the bitmap as a PNG file.

4 Click Test to determine the results of the file compression. Compare the original file size to the compressed file size to determine if the selected compression setting is acceptable.

You can also globally adjust the compression settings for JPEG files.

To globally control bitmap compression for JPEG files:

1 Select File > Publish Settings, then select the Flash tab. The Publish Settings dialog box with the Flash tab options appears:



2 Adjust the JPEG Quality slider or enter a value.

A higher JPEG quality value results in a higher image quality. As with the compression settings previously described, lower image quality produces a smaller SWF file; higher image quality produces a larger SWF file. Try different settings to determine the best trade-off between size and quality.

Vector graphics

Whenever possible do not use borders in your vector graphics; this will greatly diminish the number of rendered lines.

Using ActionScript

Because of CPU limitations, you should adhere to the following general guidelines when developing ActionScript for Flash Lite movies deployed on i-mode phones:

- Keep the ActionScript as simple as possible.
- Limit the number of loops that you use and the amount of code that each loop contains.
- Stop frame-based looping as soon as it is no longer needed.
- Avoid string and emulated array processing—it can be extremely CPU intensive.

  Note: Flash 4 ActionScript does not support arrays. However, they can be emulated using the `eval()` function. For more information, see Macromedia TechNote 14219, "How to use Eval to emulate an array," at www.macromedia.com/go/flash_support.

Interactive versus Inline content

Flash Lite movies can be viewed in the i-mode browser in one of two modes: Interactive or Inline. The browser determines the mode used.

In Interactive mode, the user can view and interact with the Flash Lite movie. Generally, this occurs when a Flash Lite movie is loaded directly into the browser without being embedded in an i-mode compatible HTML web page or mixed with any other type of content. The browser then sends all supported key events to Flash Lite, allowing content to access the network and play sound.

If the movie's display area is not the same as the browser's display area, the browser re-sizes the movie to fit the browser's display area. No horizontal or vertical scrolling is required or possible. The movie's aspect ratio does not change.

Inline mode occurs when a Flash Lite movie is embedded in an i-mode compatible HTML page that contains another Flash Lite movie or other HTML controls and objects. The browser does not send any key events to Flash Lite, eliminating the possibility of interactivity. Because the `getURL()` function and event sound is only triggered by keypress events, Inline Flash movies cannot access the network or play sound.

For Inline mode, the movie's display size can be larger than the browser's display area. The movie is scaled so the movie's width does not exceed the browser's width. However, the movie's height may end up being larger than the browser's height, in which case the entire browser page can be scrolled vertically.

CHAPTER 3
Sound

Embedding sound in Flash Lite movies for i-mode phones

The *Macromedia Flash MX Professional 2004 User Guide for Flash Lite* describes the general process and tools required to embed sound in Flash Lite movies running on a phone. This chapter contains additional information, including procedures, specific to embedding sound in Flash Lite movies for 505i phones.

Flash Lite content for 505i phones supports only the Melody Format for i-mode (MFi) audio format. Each manufacturer's 505i phone supports the standard MFi format, plus its own proprietary extensions.

MFi sound substitution

Flash Lite does not support standard Flash Player audio formats—Raw, ADPCM, or MP3. For 505i phones, only the MFi (Melody Format for i-mode) audio format is supported. Since Flash MX Professional 2004 does not natively support MFi, you must temporarily substitute a proxy sound in a recognized format such as MP3. You can use options in the Sound Properties dialog box and the Flash Publish Settings dialog box to link the proxy sound file to an MFi sound file.

Review the *Macromedia Flash MX Professional 2004 User Guide for Flash Lite* to understand the basics of sound substitution. The examples and tutorials in the user guide use MIDI (Musical Instrument Digital Interface) sound files, but the same principles apply for MFi files.

The following procedure is specific to i-mode phones.

Associating an MFi sound file with an ActionScript sound symbol

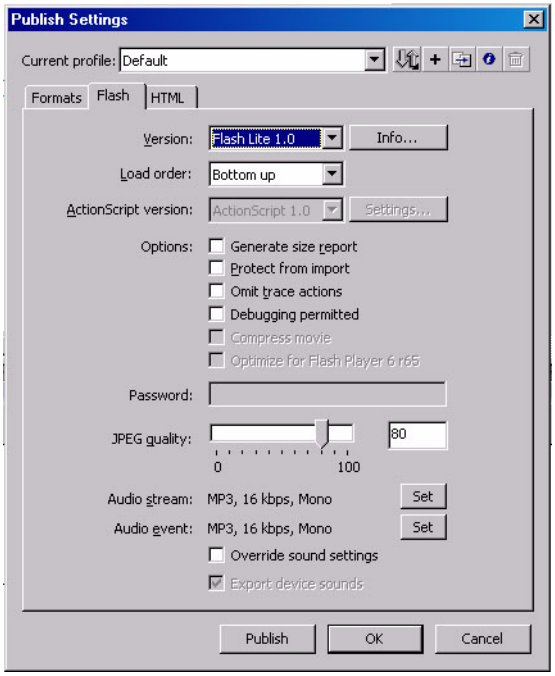
This procedure illustrates a simple case of associating an MFi sound file with an ActionScript sound symbol so the Flash MX Professional 2004 test movie player can recognize and play it.

To associate an MFi file with an ActionScript symbol:

1 In your sound authoring program, create an MFi sound file and save it as **MySound.mld**.

2 In Flash MX Professional 2004, create a new file and name it **FlashLiteSound.fla**. Save it in the same directory as MySound.mld.

3 Select File > Publish Settings > Flash tab. The Publish Settings dialog box appears:



- In the Version pop-up list, select Flash Lite 1.0.
- Click OK.

4 Select Window > Other Panels > Common Libraries > Buttons. Select a button and drag it to the Stage.

5 Double-click the new button. The Timeline should change to edit the button and display frames named Up, Over, Down, and Hit.

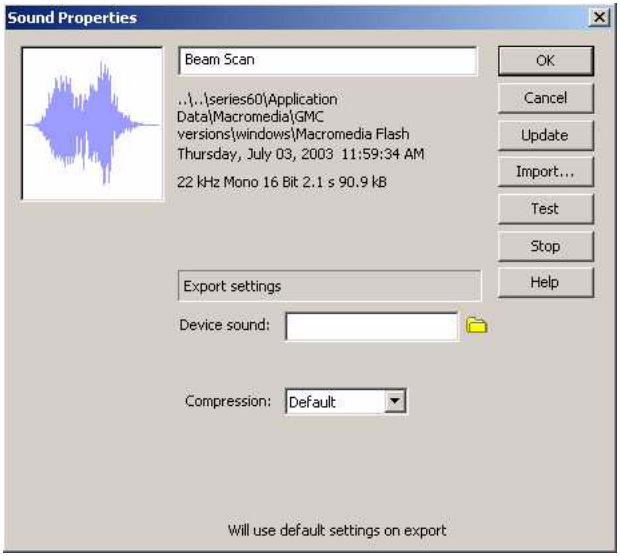
6 Select Insert > Timeline > Layer to create a new layer. Select Modify > Timeline > Layer Properties and change the name of the layer to **Sound**.

7 Select the Down frame in the Sound layer and insert a keyframe.

8 Select a sound from the Sounds library window and drag it to the keyframe.

9 Associate the sound with the MySound.mld file by doing the following:

- Select Window > Library and find the sound that you added earlier. Select the sound and right-click it to open the context menu. Select Properties from the context menu. The Sound Properties dialog box appears:



- For the Device sound option, use the file browser to find and select MySound.mld.
- Click OK.

10 Select Control > Test Movie to start the Flash Lite 1.0 test movie player.

11 Click in the test movie player window. Since Flash Lite ignores the mouse, press the Tab key until the focus is on the button. (You may need to select Control -> Disable Keyboard Shortcuts before you can navigate with the Tab key.) Press Enter. You should hear the sound from the MFi file you created in step 1.

12 To playback Flash movies that contain sound data in MFi, use Flash Lite 1.0 test movie or the Standalone Flash Lite Player (SAFlashLite).

Accessing sound on main and movie clip Timelines

Sounds do not necessarily need to reside on a button Timeline. In fact, there may be times when it is useful to access sound that resides on either the main Timeline or a movie clip Timeline. As described earlier, to successfully playback any sound it must be attached to a button, but with this method the frame on the main or movie clip Timeline that contains the sound symbol must be called using the `gotoAndPlay` action within the `keyPress` event statement.

This procedure illustrates a simple case of associating an MFi sound file with an ActionScript sound symbol on the main or movie clip Timeline so the Flash Lite test movie player can recognize and play it.

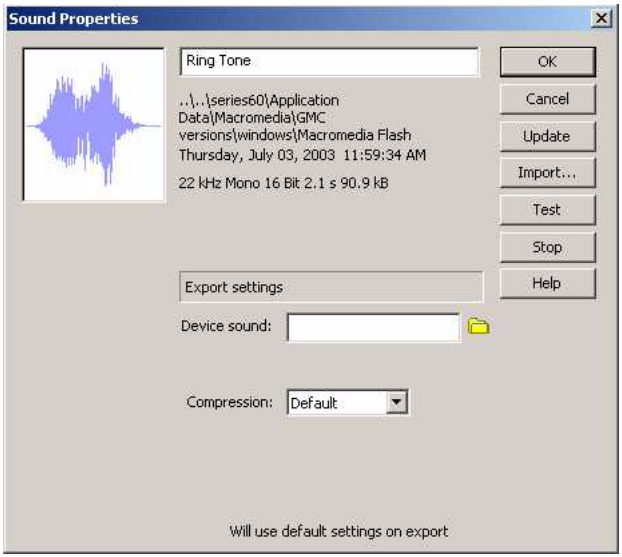
To associate an MFi file with an ActionScript symbol on a main or movie clip Timeline:

1. In your sound authoring program, create an MFi sound file and save it as **MySound.mld**.
2. In Flash MX Professional 2004, create a new file and name it **FlashLiteSound.fla**. Save it in the same directory as MySound.mld.
3. Select File > Publish Settings > Flash tab.
   - In the Version pop-up menu, select Flash Lite 1.0 and click OK.
   - Click OK.
4. Select Window > Other Panels > Common Libraries > Buttons. Select a button and drag it to the Stage.
5. Select Insert > Timeline > Layer to create a new layer on the main Timeline. Select Modify > Timeline > Layer Properties and change the name of the layer to **Sound**.
6. Click on the new **Sound** layer in the main Timeline and select Insert > Keyframe to add a new Keyframe to the **Sound** layer.
7. Select Window > Other Panels > Common Libraries > Sounds to open the Sounds Library window.
8. Select Window > Library to open the current document's Library window.
9. Select a sound in the Sounds Library window and drag it to the document Library window.
10. Associate the sound with the second keyframe in the **Sound** layer:
    - Select the sound from the Sound pop-up menu in the Properties inspector. (Select Windows > Properties to display the Properties inspector if it is not already visible.)

      Note: The sound may not appear immediately in the pop-up menu. You may have to select another frame and then reselect the Down frame to get the sound to appear in the pop-up menu.

11 Link the sound with MySound.mld:

If the Library window is not already open, select Window > Library and find the sound that you added earlier. Select the sound and right-click it to open the context menu. Select Properties from the context menu. The Sound Properties dialog box appears:



- For the Device sound option, use the file browser to find and select MySound.mld.
- Click OK.

12 Select Insert > Timeline > Layer to create a new layer on the main Timeline. Select Modify > Timeline > Layer properties and change the name of the layer to **Actions**.

13 Select Insert > Timeline > Keyframe to add a new Keyframe to the new **Actions** layer.

14 Click on the first Keyframe on the **Actions** layer and in the Actions window enter the following script:

```
stop();
```

Note: If the Actions window is not already open you will need to select Window › Development Panels › Actions

15 Click on the button you added to the stage and in the Actions window add the following script:

```
on(keyPress "1"){
  gotoAndPlay(2);
}
```

16 Insert a Keyframe in the **Actions** layer where you would like the sound to stop playing. For example: 200. Select this Keyframe and in the Actions window add the following script:

```
stop();
stopAllSounds();
```

17 Select Control > Test Movie to start the test movie player.

18 Click in the test movie player window and press the "1" key. (You may need to select Control > Disable Keyboard Shortcuts before you can execute `keyPress` actions.) You should hear the sound from the MFi file you created in step 1.

Note: You can also access sound that resides in movie clip Timelines in much the same manner except you will need to use the tellTarget action in coordination with the gotoAndPlay() action. Here is an example of a script which targets a sound in a movie clip Timeline:

```
On(keyPress “1”){
  TellTarget(“myMovieClip”){
    GotoAndPlay(2);
  }
}
```

CHAPTER 4
Testing Content

Test your Flash Lite movies frequently on actual 505i phones. This advice may sound obvious, but this step is often overlooked and is especially important for developing Flash Lite movies for i-mode phones. No matter how much phone emulation a developer does, the final delivery remains the most important step in the development cycle. Emulation is helpful for much of the testing, but it is no substitute for testing on actual 505i phones.

For basic information on how to use Flash MX Professional 2004 to author and preview Flash Lite movies created for playing on phones, please refer to the *Macromedia Flash MX Professional 2004 User Guide for Flash Lite* .

You should use the following to test your Flash Lite movie for i-mode phones:

- The test movie Flash Lite Player (invoked during the Test Movie process)
- The stand-alone Flash Lite simulator
- The i-mode HTML Simulator from DoCoMo
- Flash Lite on the manufacturer's i-mode phone

The Macromedia Flash MX Professional 2004 test movie player recognizes and plays FlashLite movies. When you select Control > Test Movie or Control > Test Scene, new information, warning, and error messages specifically related to Flash Lite movies are displayed in a separate Output window.

Whenever an unknown tag is encountered, warning messages are displayed so that the author can modify the content appropriately. Not all invalid Flash content is flagged as being in error, such as invalid ActionScript and key input.

For a detailed explanation of all messages related to Flash Lite, see Appendix C, "Warning and Error Messages" on page 41. This appendix lists all of the warning and error messages that you might see when creating Flash Lite movies for i-mode phones.

DoCoMo's i-mode HTML Simulator

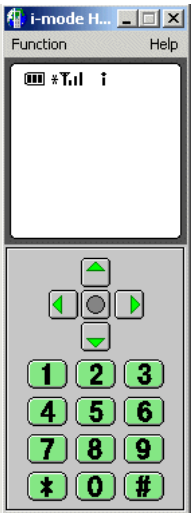
DoCoMo provides an i-mode HTML Simulator for testing Flash Lite movies on the desktop. The i-mode HTML Simulator is available from the DoCoMo website (see Appendix D, "References," on page 47).

Note: There is no i-mode HTML Simulator application for the Macintosh.

To use the i-mode HTML Simulator:

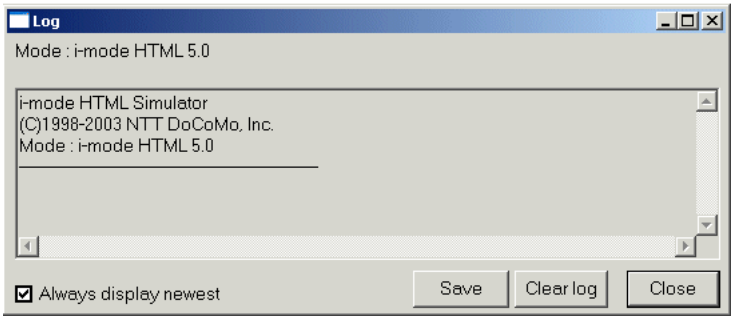
1 Download and install the i-mode HTML Simulator application from the DoCoMo website (for a link to the website, see Appendix D, "References," on page 47). Follow the online instructions. Note the folder where the program files are installed.

2 Start the i-mode HTML Simulator application. In Windows Explorer, or another program displaying filenames or icons, go to the folder where the Simulator program files are installed (noted in step 1) and double-click CSim.exe. The i-mode HTML Simulator application window appears:



You can click the Simulator keys with your mouse and the keys will operate in the same way as they would on an actual i-mode phone. The Select key is the key with a circle icon, surrounded by the arrow keys, immediately under the screen display.

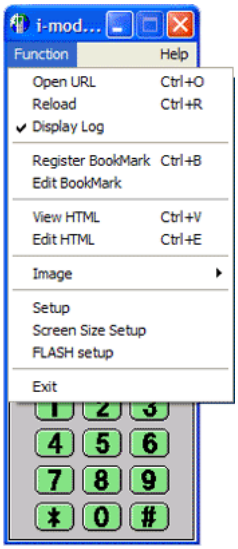
At the same time as the Simulator application window appears, a Log output window also appears:



The Log window records your interactions with the Simulator and displays information, warning, and error messages regarding your tested files. Choose Function > Display in the Simulator application to open or close the Log window.

3 Review the i-mode HTML Simulator functions:

- Click Function at the top of the Simulator application window. The following pop-up menu appears:



The following function menu options for testing your Flash Lite movies are available:

Open URL	Enables you to type in the URL of the file you want the Simulator to test. You can also browse for a file on your local computer or use a bookmark. (A bookmark is a shortcut for a file URL or pathname.)
Reload	Retest and reload the file you previously opened using the Open URL menu option.
Display Log	Show or hide the Log window.
Register BookMark	Save the location of the currently opened file and give it a shortcut name so you can readily load the file again at a later time using Open URL.
Edit BookMark	Edit the location and shortcut name of a bookmark.
View HTML	View the i-mode compatible HTML of your currently loaded test file.
Edit HTML	Edit the i-mode compatible HTML of your currently loaded test file in Notepad or another text editor.
Image	Copy to the clipboard or print the current screen display.
Setup	Set the i-mode compatible HTML version you want your test file to be verified against.
Screen Size Setup	Specify the width and height of your screen display area and set the display font size.
FLASH setup	Set the run-time memory available to Flash Lite movies running in the i-mode HTML simulator.
Exit	Exit from the Simulator application.

APPENDIX A
Supported ActionScript

This appendix lists the Flash Lite ActionScript commands and any exceptions to the standard in Flash Lite for i-mode.

Action Name	Description	Support
// (comment)	Comment; indicates the beginning of a script comment. Any characters that appear between the comment delimiter // and the end-of-line character are interpreted as a comment.	Fully supported
, (comma)	Operator; a separator between two expressions that causes the value of the second expression to be the return value.	Fully supported
. (dot)	Operator; used to navigate movie clip hierarchies in order to access nested (child) movie clips, variables, or properties.	Fully supported
" " (string delimiter)	String delimiter; when used before and after characters, quotes indicate that the characters have a literal value and are considered a string-not a variable, numerical value, or other ActionScript element.	Fully supported
- - (decrement)	Operator; a pre-decrement and post-decrement unary operator that subtracts one(1) from an expression.	Fully supported
++ (increment)	Operator; a pre-increment and post-increment unary operator that adds 1 to an expression.	Fully supported
+ (add)	A numeric operator used for adding numbers.	Fully supported

+= (addition assignment)	Operator (arithmetic); assigns to expression1 the value of expression1 + expression2
	For example, the following two statements have the same result:
	x += y;
	x = x + y;
		Fully supported

Action Name	Description	Support
(-) subtract	Operator (arithmetic); used for negating or subtracting. When used for negating, it reverses the sign of the numerical expression. When used for subtracting, it performs an arithmetic subtraction on two numerical expressions, subtracting expression2 from expression1. Example 1: The following statement reverses the sign of the expression 2 + 3. -(2 + 3) The result is -5. Example 2: The following statement subtracts the integer 2 from the integer 5. 5 - 2 The result is 3.	Fully supported

-= (subtraction assignment)	Operator (arithmetic); assigns to expression1 the value of expression1 - expression2
	
	For example, the following two statements have the same result:
	x -= y;
	x = x - y;
		Fully supported

* (multiply)	Operator (arithmetic); multiplies two numerical expressions.	Fully supported

*= (multiplication assignment)	Operator (arithmetic); assigns to expression1 the value of expression1 * expression2
	
	For example, the following two expressions are the same:
	x *= y;
	x = x * y;
		Fully supported

/ (divide)	Operator (arithmetic); divides expression1 by expression2.
	
	For example, the following statement sets the value of x to 25:
	y = 50;
	x = y/2;
		Fully supported

/= (division assignment)	Operator (arithmetic); assigns to expression1 the value of expression1 / expression2
	
	For example, the following two statements are the same:
	x /= y;
	x = x / y;
		Fully supported

= (numeric equality)	A numeric equality operator used to test two expressions for equality. The result is true if the expressions are equal.	Fully supported

Action Name	Description	Support

< (less than)	Operator (comparison); compares two expressions and determines whether expression1 is less than expression2 (true), or whether expression1 is greater than or equal to expression2 (false). In Flash Lite (and Flash 4), < is a numeric operator and is only used for expressions and not strings.
	
	The following examples illustrate true and false returns for < comparisons.
	3 < 10;
	// true
	
	10 < 3;
	// false
		Fully supported

<= (less than or equal to)	Operator (comparison); compares two expressions and determines whether expression1 is less than or equal to expression2 (true), or whether expression1 is greater than expression2 (false).
	
	The following examples illustrate true and false results for <= comparisons:
	5 <= 10;
	// true
	
	2 <= 2;
	// true
	
	10 <= 3;
	// false
		Fully supported

> (greater than)	Operator (comparison); compares two expressions and determines whether expression1 is greater than expression2 (true), or whether expression1 is less than or equal to expression2 (false).
	
	The following examples illustrate true and false returns for > comparisons.
	10 > 3;
	// true
	
	3 > 10;
	// false
		Fully supported

Action Name	Description	Support

>= (greater than or equal to)	Operator (comparison); compares two expressions and determines whether expression1 is greater than or equal to expression2 (true), or whether expression1 is less than expression2 (false).
	
	The following examples illustrate true and false results for >= comparisons:
	10 >= 5;
	// true
	
	2 >= 2;
	// true
	
	3 >=10;
	// false
		Fully supported

<> (inequality)	Operator (equality); tests the opposite of the equality operator. If expression1 is equal to expression2, the result is false.
	The following examples illustrate true and false returns for the <> operator.
	3 < > 10;
	// true
	
	3 <> 3;
	// false
		Fully supported

% (modulo)	Operator; calculates the remainder of expression1 divided by expression2. For example, the following statement sets the value of x to 3: x = 45 % 6;	Fully supported
%= (modulo assignment)	Operator (assignment); assigns to expression1 the value of expression1 % expression2. For example, the following two expressions are the same: x %= y x = x % y	Fully supported

Action Name	Description	Support

|| (logical OR)	Operator (logical); evaluates expression1 and, if expression1 is false, evaluates expression2. The result is (true) if either or both expressions evaluate to true; the result is (false) only if both expressions evaluate to false.
	
	The following example uses the || operator in an if statement. The second expression evaluates to true so the final result is true:
	x = 10;
	y = 250;
	if (x > 25 || y > 200) {
	z = 5;
	}
	else {
	z=0;
	}
	// z has a value of 5 after the code above has executed
		Fully supported

! (logical not)	Operator (logical); inverts the Boolean value of a variable or expression.	Fully supported

&& (logical AND)	Operator (logical); evaluates expression1 and, if expression1 is true, evaluates expression2. The result is (true) if both expressions evaluate to true; the result is (false) if either expression evaluates to false.
	
	The following example uses the && operator in an if statement. Both expressions evaluate to true, so the final result is true:
	x = 30;
	y = 250;
	if (x > 25 && y > 200) {
	z = 5;
	}
	else {
	z = 0;
	}
	// z has a value of 5 after the code above has executed
		Fully supported

?: (conditional)	Operator (conditional); evaluates expression1, and returns the value of expression2 if expression1 is true; otherwise, returns the value of expression3.
	
	The following statement assigns the value of variable x to variable z because expression1 evaluates to true:
	x = 5;
	y = 10;
	z = (x < 6) ? x : y;
	// z has a value of 5
		Fully supported

Action Name	Description	Support
& (string concatenation)	Operator; used for concatenating strings.	Fully supported
add	Operator; concatenates (combines) two or more strings.	Fully supported
and	Operator; performs a logical AND operation. If both expressions evaluate to true, then the entire expression is true.	Fully supported
break	Action; appears within a loop (for, for...in, do...while or while). The break action skips the rest of the loop body, stopping the looping action, and executes the statement following the loop statement. Use the break action to break out of a series of nested loops.	Fully supported
call	Action; switches the context from the current script to the script attached to the frame being called.	Fully supported
case	Keyword; defines a condition for the switch action.	Fully supported
chr()	String function; converts ASCII code numbers to characters.	Fully supported
continue	Action; used to control code execution in loops.	Fully supported
do... while	Action; executes the statements inside the loop, and then evaluates the condition of the loop for as long as the condition is true.	Fully supported
duplicateMovieClip	Action; creates an instance of a movie clip while the movie is playing.	Fully supported
else	Action; specifies the actions, clauses, arguments, or other conditional to run if the initial if statement returns false.	Fully supported
else if	Action; evaluates a condition and specifies the statements to run if the condition in the initial if statement returns false.	Fully supported

eq (string equal)	Comparison operator; compares two expressions for equality and returns true if expression1 is equal to expression2; otherwise, returns false. This action is string specific.
	
	The following examples illustrate true and false results for the eq operator:
	x ="Amy";
	y="Fred";
	x eq "Amy";
	// true
	x eq y;
	// false
		Fully supported

eval()	Function; accesses variables - the value of the variable is returned.	Fully supported

Action Name	Description	Support
fscommand()	Action; allows the Flash movie to communicate with the program hosting Flash Lite.	Not supported

ge (string greater than or equal)	Comparison operator; returns true if the string representation for expression1 is greater than or equal to the string representation for expression2; otherwise, returns false. This action is string specific.
	
	The following examples illustrate true and false results for the ge operator:
	x =”Amy”;
	y=”Fred”;
	x ge y;
	// false
	x ge “Amy”;
	// true
	y ge x;
	// true
		Fully supported

getProperty()	Function; returns the value of the specified property for the movie clip instance.	Partially supported. Not all properties are supported - see Appendix B, “Supported Properties,” on page 37.
getTimer()	Function; returns the number of milliseconds that have elapsed since the movie started playing.	Fully supported
getURL()	Action; loads a document from a specific URL into a window or passes variables to another application at a defined URL. When sending variables, specify whether to load variables using a GET or POST method. GET appends the variables to the end of the URL, and is used for small numbers of variables. POST sends the variables in a separate HTTP header and is used for long strings of variables.	Partially supported. The URL protocols http, https, mailto, and tel are supported, once per event action.
gotoAndPlay()	Action; sends the playhead to the specified frame in a scene and plays from that frame. If a scene is not specified, the playhead goes to the specified frame in the current scene.	Fully supported
gotoAndStop()	Action; sends the playhead to the specified frame in a scene and stops it. If no scene is specified, the playhead is sent to the frame in the current scene.	Fully supported
gt (string greater than)	Comparison operator; returns true if the string representation for expression1 is greater than the string representation for expression2; otherwise, returns false. This action is string specific.	Fully supported

Action Name	Description	Support
if	Action; evaluates a condition to determine the next action in a movie. If the condition is true, Flash runs the statements that follow.	Fully supported
ifFrameLoaded()	Action; checks whether the contents of a specific frame are available locally. Use ifFrameLoaded to start playing a simple animation while the rest of the movie downloads	Fully supported
int()	Function; converts a decimal number to the closest integer value.	Fully supported

le (string less than or equal)	Comparison operator; returns true if the string representation for expression1 is less than or equal to the string representation for expression2; otherwise, returns false. This action is string specific.
	
	The following examples illustrate true and false results for the le operator:
	x =”Amy”;
	y=”Fred”;
	y le x;
	// false
	x le “Amy”;
	// true
	x le y;
	// true
		Fully supported

length()	String function; returns the length of the specified string or variable name.	Fully supported
loadMovie()	Action; plays additional movies without closing Flash Lite. Normally, Flash Lite displays a single Flash movie (SWF file) and then closes. The loadMovie() action lets you display several movies at once or switch between movies without loading another HTML document.	Not supported
loadMovieNum()	Action; loads a SWF into a level in Flash Lite while the originally loaded movie is playing.	Not supported
loadVariables()	Action; reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or Personal Home Page (PHP), and sets the values for variables in a movie or movie clip.	Not supported
loadVariablesNum()	Action; reads data from an external file, such as a text file or text generated by a CGI script, Active Server Pages (ASP), or PHP, or Perl script, and sets the values for variables in a Flash Lite level.	Not supported

Action Name	Description	Support

lt(string less than)	Operator (comparison); compares expression1 to expression2 and returns true if expression1 is less than or equal to expression2; otherwise, returns false. This action is string specific.
	
	The following examples illustrate true and false results for the lt operator:
	x ="Amy";
	y="Fred";
	y lt x;
	// false
	x lt "Jane";
	// true
		Fully supported

mbchr()	String function; converts an ASCII code number to a multibyte character.	Fully supported
mblength()	String function; returns the length of the multibyte character string.	Fully supported
mbord()	String function; converts the specified character to a multibyte number.	Fully supported
mbsubstring()	String function; extracts a new multibyte character string from a multibyte character string.	Fully supported

ne (string not-equal)	Comparison operator; compares two expressions for inequality and returns true if expression1 is not equal to expression2; otherwise, returns false. This action is string specific.
	
	The following examples illustrate true and false results for the ne operator:
	x ="Amy";
	y="Fred";
	y ne"Amy";
	// true
	x ne "Amy";
	// false
		Fully supported

nextFrame()	Action; sends the playhead to the next frame and stops it.	Fully supported
nextScene()	Action; sends the playhead to frame 1 of the next scene and stops it.	Fully supported
Number()	Function; converts the argument x to a number and returns a value as follows: If x is a number, the return value is x. If x is a Boolean, the return value is 1 if x is true, 0 if x is false. If x is a string, the function attempts to parse x as a decimal number with an optional trailing exponent, that is, 1.57505e-3. If x is undefined, the return value is 0.	Not supported

Action Name	Description	Support
on(event)	Handler; specifies the mouse event, or keypress that trigger an action.	Partially supported. Events supported are keyPress, press, release, rollOver and rollout. Keys supported in Flash Lite for i-mode are: 0-9, *, # and Select.
ord()	String function; converts characters to ASCII code numbers.	Fully supported
play()	Action; moves the playhead forward in the Timeline.	Fully supported
prevFrame()	Action; sends the playhead to the previous frame and stops it.	Fully supported
prevScene()	Action; sends the playhead to frame 1 of the previous scene and stops it.	Fully supported
random()	Function; returns a random integer between 0 and the integer specified in the value argument.	Fully supported
removeMovieClip()	Action; deletes a movie clip instance that was created with the duplicateMovieClip action.	Fully supported
set()	Action; assigns a value to a variable. A variable is a container that holds information.	Fully supported
setProperty()	Action; changes the property of a movie clip as the movie plays.	Partially supported. Not all properties are supported - see Appendix B, "Supported Properties," on page 37
startDrag()	Action; makes the target movie clip draggable while the movie is playing. Only one movie clip can be dragged at a time.	Not supported
stop()	Action; stops the movie that is currently playing.	Fully supported
stopAllSounds()	Action; stops all sounds currently playing in a movie without stopping the playhead.	Fully supported
stopDrag()	Action; stops the current drag operation.	Not supported

Action Name	Description	Support
String()	Function; returns a string representation of the specified argument as follows: If *x* is Boolean, the return string is true or false. If *x* is a number, the return string is a decimal representation of the number. If *x* is a string, the return string is x. If *x* is a movie clip, the return value is the target path of the movie clip in slash (/) notation. If *x* is undefined, the return value is an empty string.	Not supported
substring()	String function; extracts part of a string.	Fully supported
switch()	Action; creates a branching structure for ActionScript statements. The switch action tests a condition and executes statements if the condition returns a value of true.	Fully supported
tellTarget()	Action; Can be used to apply instructions to a particular Timeline or movie clip. For example, tellTarget can be assigned to buttons that stop or start movie clips on the Stage or prompt movie clips to jump to a particular frame.	Fully supported
toggleHighQuality()	Action; turns anti-aliasing on and off in Flash Lite. Antialiasing smooths the edges of objects but results in slower movie playback. The toggleHighQuality action affects all movies in Flash Lite.	Fully supported
trace()	Action; evaluates the expression and displays the results in the Output window in test movie mode.	Fully supported
unloadMovie()	Action; removes a movie from Flash Lite that was previously loaded or created using the loadMovie or duplicateMovieClip actions.	Fully supported
unloadMovieNum()	Action; removes a movie at a specified level from Flash Lite that was previously loaded or created using the loadMovie action.	Fully supported
while()	Action; runs a statement or series of statements repeatedly in a loop as long as the condition argument is true.	Fully supported

APPENDIX B
Supported Properties

This appendix lists the Flash Lite ActionScript properties and whether there are any exceptions in the Flash List for i-mode.

Properties	Description	Support
/	Property; specifies or returns a reference to the root movie Timeline. Functionality provided by this property is similar to that provided by the `_root` property in Flash 5.	Fully supported
:	Used in conjunction with "/" to reference variables and properties of other movie clips that are contained in the current movie. It is also used with the "Call" action to reference a frame label of a movie clip.	Fully supported
_alpha	Property; sets or retrieves the alpha transparency (*value*) of the movie clip. Valid values are 0 (fully transparent) to 100 (fully opaque).	Fully supported
_currentframe	Property (read-only); returns the number of the frame where the playhead is currently located in the Timeline.	Fully supported
_droptarget	Property (read-only); returns the absolute path in slash syntax notation of the movie clip instance on which the *draggableInstanceName* (the name of a movie clip instance that was the target of a startDrag action) was dropped. This property always returns a path that starts with /.	Not supported
_focusrect	Property (global); specifies whether a yellow rectangle appears around the button that has the current focus. The default value true (nonzero) displays a yellow rectangle around the currently focused button or text field as the user presses the Tab key to navigate.	Fully supported

Properties	Description	Support
_framesloaded	Property (read-only); the number of frames that have been loaded from a streaming movie. This property is useful for determining whether the contents of a specific frame, and all the frames before it, have loaded and are available locally in a user's browser.	Fully supported
_height	Property (read-only); retrieves the height of the space occupied by a movie's content. In Flash Lite, `_height` is a read-only property.	Fully supported
_highquality	Property (global); specifies the level of anti-aliasing applied to the current movie. This property can be used to control bitmap smoothing as well.	Partially supported (Bitmap smoothing not supported)
_level	In Flash Lite, movies are assigned a number according to the order in which they were loaded. The movie that was loaded first is loaded at the bottom level, level 0. The movie in level 0 sets the frame rate, background color, and frame size for all subsequently loaded movies. Movies are then stacked in higher numbered levels above the movie in level 0. This property is a reference to the root movie Timeline of *levelN*.	Fully supported
Maxscroll	Property; a read-only property that works with the scroll property to control the display of information in a text field. This property can be retrieved, but not modified.	Not supported
_name	Property; specifies the movie clip instance name.	Fully supported
_rotation	Property; specifies the rotation of the movie clip in degrees.	Fully supported
Scroll	Controls the display of information in a text field associated with a variable. The scroll property defines where the text field begins displaying content. After you set it, Flash Lite updates it as the user scrolls through the text field. The scroll property is useful for directing users to a specific paragraph in a long passage, or creating scrolling text fields.	Not supported

Properties	Description	Support
_soundbuftime	Property (global); establishes the number of seconds of streaming sound to prebuffer.	Not supported
_target	Property (read-only); returns the target path of the movie clip instance specified as argument.	Fully supported
_totalframes	Property (read-only); evaluates the movie clip specified as argument and returns the total number of frames in the movie.	Fully supported
_url	Property (read only); retrieves the URL of the SWF file from which the movie clip was downloaded.	Not supported
_visible	Property; determines whether the specified movie clip is visible. Movie clips that are not visible (property set to false) are disabled.	Fully supported
_width	Property (read-only); retrieves the width of the space occupied by a movie's content. In Flash Lite, _width is a read-only property.	Fully supported
_x	Property; sets the *x* coordinate of movie relative to the local coordinates of the parent movie clip.	Fully supported
_xscale	Property; determines the horizontal scale (percentage) of the movie clip as applied from the registration point of the movie clip.	Fully supported
_y	Property; sets the *y* coordinate of movie relative to the local coordinates of the parent movie clip.	Fully supported
_yscale	Property; sets the vertical scale (percentage) of the movie clip as applied from the registration point of the movie clip.	Fully supported

APPENDIX C
Warning and Error Messages

This appendix lists the possible information, warning, and error messages you might encounter when creating movies for Flash Lite for i-mode.

Message Identifier	Message	Explanation
SWFS016	Detected loadMovie() - will be ignored.	The Flash player detected that the SWF movie contains a `loadMovie()` ActionScript command, which the specified device's Flash Lite does not support. No modifications are made to the device-specific SWF file - this is just a warning.
SWFS017	Detected loadVariables() – will be ignored.	The Flash player detected that the SWF movie contains a `loadVariables()` ActionScript command, which the specified device's Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS018	Detected getURL() - restrictions may apply.	The Flash player detected that the SWF movie contains a `getURL()` ActionScript command, which has some run-time restrictions when played by the specified device's Flash Lite. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS019	startDrag() action not supported.	The Flash player detected that the SWF movie contains a `startDrag()` ActionScript command, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS020	stopDrag() action not supported.	The Flash player detected that the SWF movie contains a `stopDrag()` ActionScript command, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.

Message Identifier	Message	Explanation
SWFS021	_droptarget property not supported.	The Flash player detected that the SWF movie contains a `getProperty()` or `setProperty()` ActionScript command referring to the `_droptarget` property, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS023	_soundbuftime property not supported.	The Flash player detected that the SWF movie contains a `getProperty()` or `setProperty()` ActionScript command referring to the `_soundbuftime` property, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS024	scroll property not supported.	The Flash player detected that the SWF movie contains an ActionScript reference to the `scroll` property, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS025	maxscroll property not supported.	The Flash player detected that the SWF movie contains an ActionScript reference to the `maxscroll` property, which Flash Lite does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS027	File saved as <filename>	The Flash player displays this message to indicate the name it is using for the device-specific SWF file.
SWFS028	File size after substitution: <nnn> kilobytes	The Flash player displays this message to indicate the size of the device-specific SWF file after substitution or removal of sounds. This is an informational message only.
SWFS032	Detected fscommand() - will be ignored.	The Flash player detected that the SWF movie contains a `fscomamnd()` ActionScript command, which Flash Lite for i-mode does not support. No modifications are made to the device-specific SWF file – this is just a warning.
SWFS033	Not enough memory to perform operation.	The Flash player was unable to get enough memory to finish the operation
SWFS034	Input Text Fields not supported.	The Flash player detected that the SWF movie contains an input text field, which is not supported by -this is just a warning

Message Identifier	Message	Explanation
SWFS035	_url property not supported.	The Flash player detected that the SWF movie contains a getProperty or setProperty Actionscript command referring to the _url property, which is not supported by Flash Lite. No modifications will be made to the device specific SWF file - this is just a warning
SWFS036	Detected loadMovie - restrictions may apply.	The Flash player detected that the SWF movie contains a loadMovie ActionScript command, which has some runtime estrictions when played by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning.
SWFS037	Detected loadVariables - restrictions may apply.	The Flash player detected that the SWF movie contains a loadVariables ActionScript command, which has some runtime restrictions when played by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning.
SWFS038	Detected FSCommand - restrictions may apply.	The Flash player detected that the SWF movie contains a FSCommand ActionScript command, which has some runtime restrictions when played by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning.
SWFS039	Detected getURL - will be ignored.	The Flash player detected that the SWF movie contains a getURL ActionScript command, which is not supported by the specified device's Flash Player. No modifications will be made to the device specific SWF file - this is just a warning.
SWFS040	UnCompressed Sound found.	The Flash player detected that the SWF movie contains uncompressed sound. which is not supported by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning.
SWFS041	ADPCM Sound found.	The Flash player detected that the SWF movie contains ADPCM sound. which is not supported by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning.

Message Identifier	Message	Explanation
SWFS042	Nellymoser Sound found.	The Flash player detected that the SWF movie contains Nellymoser sound, which is not supported by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning
SWFS043	MP3 Sound found.	The Flash player detected that the SWF movie contains MP3 sound, which is not supported by the specified device's Flash Player. No modifications will be made to the device-specific SWF file - this is just a warning
SWFS044	Export tag ‹subst:sound file name› was found and ignored, Please use the Device sound feature.	The Flash player detected that the SWF movie contains a ‹subst:file name› export tag used in old Flash 6 updater, which is not supported by the Flash Lite 1.0 test movie player. The author shoud use the new Device Sound feature. No modifications will be made to the device-specific SWF file - this is just a warning
SWFS045	MIDI Sound found.	The Flash player detected that the SWF movie contains MIDI sound, which is supported by Flash Lite.
SWFS046	MFi Sound with ‹manufacturer› extension found.	The Flash player detected that the SWF movie contains MFi sound with certain manufacturer extension, which is supported by Flash Lite.
SWFS047	Unsupported device sound format found.	The Flash player detected that the SWF movie contains unsupported sound format, which is not supported by Flash Lite. No modifications will be made to the device-specific SWF file - this is just a warning
FTPE001	the key will not be processed: ‹key› keycode: ‹nnn›	While testing the movie, a key was pressed that Flash Lite for i-mode does not support – the keypress is ignored.
FTPA002	fscommand() is ignored.	While testing the movie, a `fscommand()` ActionScript command was encountered. Flash Lite for i-mode does not support this command and ignores it.
FTPA003	loadVariables() is ignored.	While testing the movie, a `loadVariables()` ActionScript command was encountered. Flash Lite for i-mode does not support this command and ignores it.
FTPA004	loadMovie() is ignored.	While testing the movie, a `loadMovie()` ActionScript command was encountered. Flash Lite for i-mode does not support this command and ignores it.

Message Identifier	Message	Explanation
FTPA005	The call to getURL() for ‹URL› was ignored because there was more than one request per keypress.	While testing the movie, multiple ActionScript `getURL()` commands were called during a keypress event. Flash Lite for i-mode only allows one `getURL()` command per keypress, so only the first command is processed – the others are ignored.
FTPA006	The call to getURL() for ‹URL› was ignored because it was not associated with a keypress.	While testing the movie, a `getURL()` ActionScript command was encountered outside of a keypress event. Flash Lite for i-mode only allows `getURL()` commands to be handled during a keypress event. Calls to `getURL()` outside of a keypress event are ignored.
FTPA007	getProperty or setProperty not supported for: ‹property name›	While testing the movie, a `getProperty()` or `setProperty()` ActionScript command was encountered for a property that the specified device's Flash player does not support. The command is ignored.
FTPA008	getProperty or setProperty not fully supported for: ‹property name›	While testing the movie, a `getProperty()` or `setProperty()` ActionScript command was encountered for a property that Flash Lite for i-mode does not completely support. The command is performed, but the results might not be as expected.
FTPA009	startDrag() and stopDrag() are not supported.	While testing the movie, a `startDrag()` or `stopDrag()` ActionScript command was encountered. Flash Lite does not support these commands and ignores them.
FTPS011	Only a single sound can be played at a time (no mixing).	While testing the movie, a sound was started while another sound was already playing. Flash Lite does not support sound mixing, so the first sound is stopped to allow the second sound to play.
FTPS012	Event sound was ignored because it was not associated with a keypress.	While testing the movie, an event sound was encountered outside of a keypress event. Flash Lite for i-mode only allows event sounds to be handled during keypress events. Event sounds outside of a keypress event are ignored.
FTPS013	Text fields are not selectable.	While testing the movie, an attempt was made to select a text field. Flash Lite for i-mode does not support Input text fields – they are rendered as non-selectable text fields.
FTPS022	ADPCM sounds not supported.	While testing the movie, an ADPCM sound was encountered. The specified device's Flash Player does not support ADPCM sound format.

Message Identifier	Message	Explanation
FTPS023	MP3 sounds not supported.	While testing the movie, an MP3 sound was encountered. The specified device's Flash Player does not support MP3 sound format.
FTPS024	MIDI/MFI sounds not supported.	While testing the movie, an MIDI/MFI sound was encountered. The specified device's Flash Player does not support MIDI/MFI sound format.
FTPS025	PCM sounds not supported.	While testing the movie, an PCM sound was encountered. The specified device's Flash Player does not support PCM sound format.
FTPS026	Debug movie is not supported in the specified test movie player	While the Flash Lite player is specfied in the publish settings, an attempt was made to debug the movie using Flash Lite 1.0 test movie player which is not supported.

APPENDIX D
References

The following websites contain further information about creating content for Flash Lite and Flash Lite for i-mode:

Macromedia websites

Information and resources for developing content for Flash Lite and Flash Lite for i-mode is available at several Macromedia websites.

- Macromedia Flash MX Professional 2004 Designer and Developer website:
  http://www.macromedia.com/go/devnet

NTT DoCoMo websites

Information on i-mode, i-mode compatible HTML, the i-mode HTML Simulator, 505i hardware characteristics, and Emoji is available at several NTT DoCoMo websites.

- NTT DoCoMo home page:
  http://www.nttdocomo.co.jp/
- i-mode general website:
  http://www.nttdocomo.co.jp/p_s/imode/
- Emoji application download website:
  http://www.nttdocomo.co.jp/p_s/imode/tag/emoji/
- Flash List for i-mode and i-mode HTML Simulator website:
  http://www.nttdocomo.co.jp/p_s/imode/flash/
- 505i hardware characteristics website:
  http://www.nttdocomo.co.jp/p_s/imode/spec/