macromedia®

FLASH®

Developing Flash Lite Applications

8

Acknowledgments

Project Management: Mary Leigh Burke

Writing: Tim Statler

Managing Editor: Rosana Francescato

Editing: Linda Adler, Mary Kraemer, Lisa Stanziano, Anne Szabla

Production Management: Patrice O'Neill, Kristin Conradi, Yuko Yagi

Media Design and Production: Adam Barnett, Aaron Begley, Paul Benkman. John Francis, Geeta Karmarkar, Masayo Noda, Paul Rangel, Arena Reed, Mario Reynoso

Special thanks to Lisa Friendly, Bonnie Loo, Erick Vera, the beta testers, and the entire Flash Lite engineering and QA teams.

First Edition: September 2005

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

Contents

Introduction 5

What's new in Flash Lite authoring 5
Guide to instructional media. 6
Additional resources 6
Typographical conventions 6

Chapter 1: Creating Interactivity and Navigation 7

Keys supported by Flash Lite 8
Using tab navigation in Flash Lite 9
Handling key events (Flash Professional Only) 13
Handling button events (Flash Professional Only) 20
Using the soft keys (Flash Professional Only) 26

Chapter 2: Working with Text and Fonts 31

About text in Flash Lite 31
Using input text fields 34
Font rendering methods in Flash Lite 38
Flash Lite rendering quality and anti-aliased text 39
Embedding font outlines in SWF files 40
Text field example application (Flash Professional Only) 42
Creating scrolling text (Flash Professional Only). 45

Chapter 3: Working with Sound 49

About sound in Flash Lite 49
Using device sound (Flash Professional Only). 50
Using native Flash sounds 56

Chapter 4: Optimizing content for performance and file size . . . 57

SWF file size and memory 57
Performance optimization 57
Device speed and frames per second 61

Chapter 5: Testing Flash Lite Content (Flash Professional Only) 63

Overview of Flash Lite testing features (Flash Professional Only) . . . 63
Using the Flash Lite emulator (Flash Professional Only) 66
Flash Lite error and warning messages (Flash Professional Only) . . 74
Selecting test devices and Flash Lite content type (Flash Professional Only) 76
Flash Lite content types (Flash Professional Only) 78
Determining platform capabilities (Flash Professional Only) 81

Appendix: Warning and Error Messages (Flash Professional Only) 85

Flash Lite emulator error and information messages (Flash Professional Only) 85

Index 93

Introduction

This manual describes how to develop applications for mobile devices using Macromedia Flash Lite. You can use various modes of navigation for different devices and how to work with text and fonts. This manual also describes how to manage the runtime memory available to Flash Lite applications running on various models of mobile phones. Using a Flash Lite emulator that is included with Macromedia Flash Professional 8, you can test and debug your application in the authoring tool before you test it on an actual device.

What's new in Flash Lite authoring

Flash Professional 8 includes the following new features to help developers create Flash Lite applications:

Flash Lite emulator The Flash Lite emulator lets you preview your content as it will function on an actual device. The emulator can configure itself to mimic the features available on any supported device. The emulator also provides debugging information that alerts you to potential problems and incompatibilities on the target device.

Device Settings dialog box You use the Device Settings dialog box to select your test devices and Flash Lite content type. When you test your content in the Flash Lite emulator, you can choose the test device you want the emulator to mimic.

Device document templates Flash Professional 8 includes document templates to let you quickly start creating content for specific devices and content types.

Guide to instructional media

The Flash Lite documentation package includes the following media to help you learn how to create Flash Lite applications:

- *Getting Started with Flash Lite* provides an overview of Flash Lite technology and developing Flash Lite content for mobile devices. It also includes a step-by-step tutorial for creating a Flash Lite application.
- *Developing Flash Lite Applications* is a comprehensive guide to creating Flash Lite content, and includes instructions for testing your applications in the integrated Flash Lite emulator.
- *Flash Lite 1.x ActionScript Language Reference* describes all the ActionScript language features available to Flash Lite developers, and provides example code.
- *Learning Flash Lite 1.x ActionScript* complements the language reference and provides additional code examples and an introduction to writing Flash 4 ActionScript, upon which Flash Lite 1.x ActionScript is based.
- The Flash Lite sample applications demonstrate key concepts and best practices discussed or mentioned in the written documentation.

Additional resources

For the latest information on developing Flash Lite applications, plus advice from expert users, advanced topics, examples, tips, and other updates, see the Mobile and Devices Developer Center at www.macromedia.com/devnet/devices/.

For TechNotes, documentation updates, and links to additional resources in the Flash Lite developer community, see the Macromedia Flash Lite Support Center at www.macromedia.com/support/flashlite/.

Typographical conventions

The following typographical conventions are used in this manual:

- *Italic font* indicates a value that should be replaced (for example, in a folder path).
- `Code font` indicates ActionScript code.
- *`Code font italic`* indicates an ActionScript parameter.
- **Bold font** indicates a verbatim entry.
- Double quotation marks (`""`) in code examples indicate delimited strings. However, programmers can also use single quotation marks.

CHAPTER 1

Creating Interactivity and Navigation

1

Macromedia Flash Lite 1.0 and Flash Lite 1.1 support user interaction through the device's keypad, or through a stylus or touch-screen interface on devices that provide one.

There are two ways to add key-based interactivity to a Flash Lite application. You can use the Flash Lite default tab navigation, or you can create a custom key-based navigation system.

Tab navigation functions the same way in Flash Lite as it does in Flash desktop applications, where the Tab and Shift+Tab keys let users navigate between objects on the screen. In Flash Lite, the device's four-way arrow keys serve the same purpose as the Tab and Shift+Tab keys. Tab navigation in Flash Lite works only with buttons and input text fields; it is typically best used for simple user interactions, such as menus. For more information, see "Using tab navigation in Flash Lite" on page 9.

Instead of using tab navigation, you can use custom key navigation. In this case, your application handles keypress events that Flash Lite generates in response to a user pressing a key on their device, and then takes the appropriate action. You would use this type of navigation, for example, if you're creating a Flash Lite game or other application whose interaction model is more complex than that of a simple menu.

This chapter contains the following topics:

Keys supported by Flash Lite 8
Using tab navigation in Flash Lite 9
Handling key events (Flash Professional Only) 13
Handling button events (Flash Professional Only) 20
Using the soft keys (Flash Professional Only) 26

Keys supported by Flash Lite

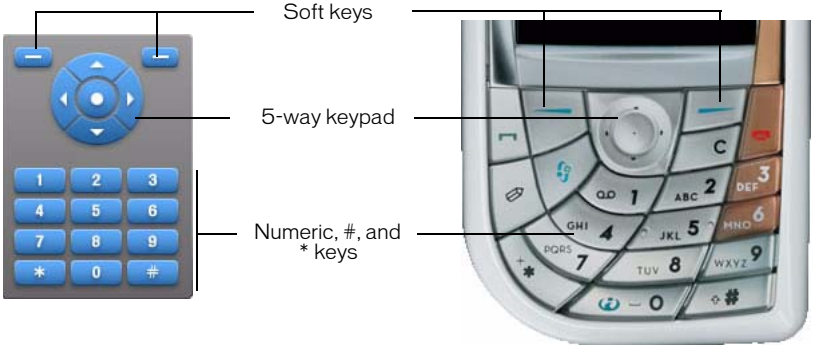
In addition to the alphanumeric keys available on standard telephones, most mobile devices feature a five-way keypad, which let users navigate and select items on the device's display as well as the Left and Right soft keys. A device's *soft keys* are multifunctional keys that use the device's display to identify their purpose at any moment.

The five-way keypad has four arrow keys (Up, Down, Left, and Right) and a Select key, which are typically located at the center of the keypad. Different applications can use these keys as they choose. For example, in a Flash Lite game, the user might press the arrow keys to move a character on the screen, and then press the Select key to perform another action, such as make the character jump.

Flash Lite supports the following keys on mobile devices:

- Five-way keypad keys (Up, Down, Left, Right, and Select)
- Left and Right soft keys
- 0-9, *, and # keys

The following images show these keys on a generic keypad and on an actual phone:



Not all devices and Flash Lite content types support all these keys. For example, on some devices, a Flash Lite application can access only the Up and Down arrow keys, not the Left and Right arrow keys. Also, not all devices and content types allow Flash applications to access the Left and Right soft keys. When you test your application in the Flash Lite emulator, the emulator generates warning messages when you press keys that aren't available on the target device and content type.

Using tab navigation in Flash Lite

On desktop Flash applications, the Tab and Shift+Tab keys let users switch keyboard focus among objects on the screen. The object that has focus responds to further keypresses. In Flash Lite, the arrow keys on the device's five-way keypad serve the same purpose as the Tab and Shift+Tab keys.

Flash Lite supports three different modes of tab navigation: two-way, four-way, and four-way with wrap-around. Different devices and Flash Lite content types support different navigation modes. In two-way navigation, users can For more information, see "Modes of tab navigation" on page 9.

Tab navigation in Flash Lite works with buttons and text fields. When an input text field has focus and the user presses the Select key, Flash Lite opens the device's generic text input dialog box, in which the user can enter text. For an example of using input text fields to get user input, see "Text field example application (Flash Professional Only)" on page 42.

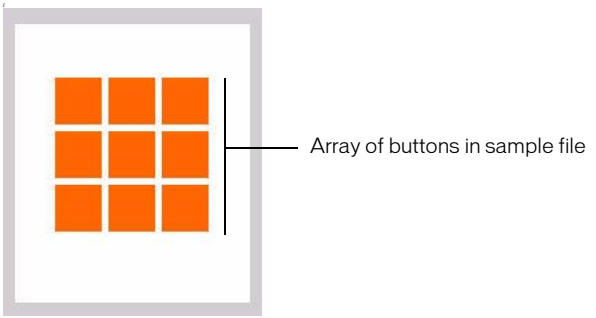
This section contains the following topics:

Modes of tab navigation 9
About the focus rectangle. 11
Guidelines for using tab navigation. 12

Modes of tab navigation

Flash Lite supports three modes of tab navigation: two-way, four-way, and four-way with wrap-around. Different devices and Flash Lite content types support different navigation modes. For information on determining the navigation mode for a specific device and content type, see "Determining platform capabilities (Flash Professional Only)" on page 81.

Each navigation mode discussed in the following sections references a sample file that you can view in the Flash Lite emulator. Each sample file consists of the same three-by-three grid of buttons, as shown below. The only difference between the sample files is that each is configured to target a device and Flash Lite content type that support the navigation mode in question.



To use each sample file, open it in Flash and test it in the emulator (select Control > Test Movie). Click the arrow keys on the emulator's keypad (or press the arrow keys on your keyboard) to see how each navigation mode affects user navigation.

Two-way navigation In two-way navigation, the device's Up and Down arrow keys move focus from one object (button or input text field) to another; the Left and Right arrow keys have no effect. The Down key moves the focus to the next object on the right. If there are no objects positioned to the right of the current object with focus, focus moves to the leftmost object below the current object with focus. If there are no objects below the rightmost object with focus, focus moves to the top and leftmost object. The Up key moves focus to the next object on the left. If there are no objects to the left of the current object with focus, focus moves to the rightmost object above the object.

For an example of two-way navigation, see the sample file named 2_way_nav.fla, located in the Samples and Tutorials/Samples/Flash Lite folder in the Flash installation folder.

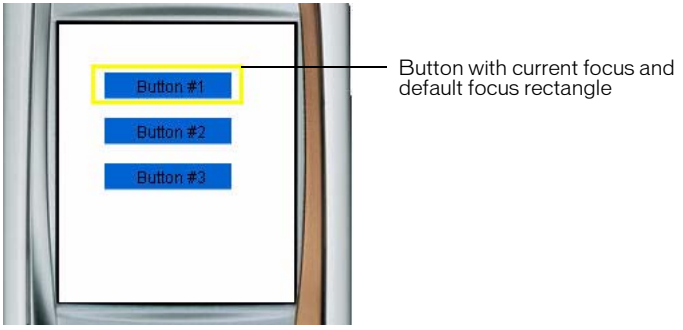
Four-way navigation In four-way navigation, the user can move focus from one object to another with the device's Left, Right, Up, and Down keys. Pressing the Left key moves the focus from the current object with focus to the object to the left of the button with focus. The Right key moves the focus to the next button to the right of the button with focus. The Up and Down keys similarly move focus to the button above and below the button with focus.

For an example that uses four-way navigation, see the sample file named 4_way_nav.fla located in the Samples and Tutorials/Samples/Flash Lite folder in the Flash installation folder.

Four-way navigation with wrap-around This mode is the same as four-way navigation, except that when there are no buttons below the rightmost button with focus, focus moves to the top and leftmost button.

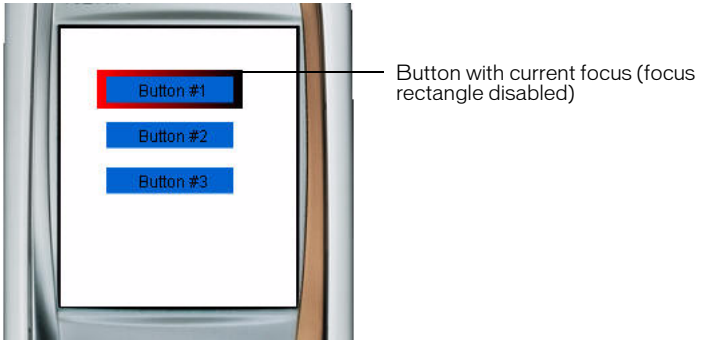
About the focus rectangle

By default, Flash Lite draws a yellow rectangle around the button or input text field that has focus. The focus rectangle lets the user know which object on the screen will respond when the user presses the device's Select key. For example, the following image shows the focus rectangle drawn around a button that has the current keypad focus:

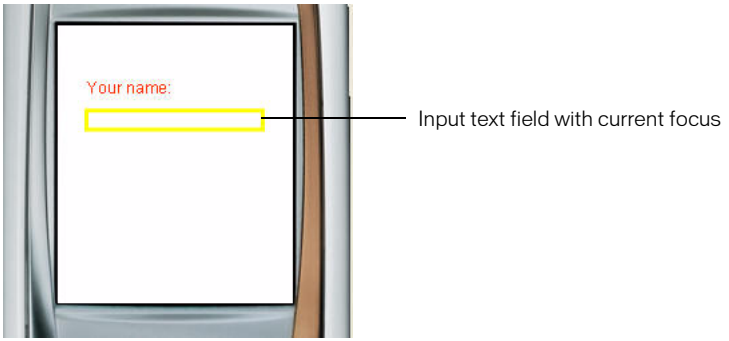


For buttons, the focus rectangle's bounding box is determined by the button's *hit area*—the invisible region that (in Flash desktop applications) defines the part of the button that responds to mouse clicks. For input text fields, the focus rectangle's bounding box is determined by the text field's dimensions.

You can disable the default focus rectangle behavior by setting the `_focusRect` property to `false`. If you're using buttons that define *over states*, Flash Lite displays those states when the button receives focus. For this reason, the focus rectangle is often not necessary when using buttons. For example, the following image shows the same application as in the preceding image, but with the focus rectangle disabled; the button that has focus is displaying its over state:



If your application contains input text fields, Macromedia recommends that you do *not* disable the focus rectangle, as it provides the only visual clue that an input text field has focus. For example, the following image shows an input text field that has the current focus:



If your application contains buttons (with defined over states) and input text fields on the same screen, you can set the `_focusRect` property to `false` in each button's `on(rollOver)` event handler and set it to `true` in each button's `on(rollOut)` handler, as shown in the following code example. This causes the focus rectangle to appear when an input text field has focus, but not when a button has focus.

```
// Attach this code to each button on the Stage.
on(rollOver) {
  _focusRect = false;
}
on(rollOut) {
  _focusRect = true;
}
```

For more information about using input text fields, see "Using input text fields" on page 34.

Guidelines for using tab navigation

When using tab navigation to create interactivity, you should place at least two objects (input text fields, buttons, or a combination) on the screen at the same time. If the screen only contains a single button or input text field, the user can't change the focus and may feel stuck in the user interface.

If a screen in your application contains only a single button for user interaction, consider detecting a keypress event rather than using button events. For more information, see "Handling key events (Flash Professional Only)" on page 13.

Handling key events (Flash Professional Only)

In addition to using tab navigation between buttons and input text fields, a Flash Lite application can also respond to arbitrary keypress events.

Not all devices and content types support all device keys. For example, on a device that supports two-way navigation (see "Modes of tab navigation" on page 9) Flash Lite doesn't generate keypress events for the Left and Right Arrow keys.

On all devices, Flash Lite supports the following keys:

- 0-9, *, and # keys
- Select key

On devices that support two-way navigation, Flash Lite also supports the Up and Down Arrow keys on the 5-way keypad. On devices that support four-way navigation, Flash Lite supports the Up, Down, Left, and Right Arrow keys.

On devices that support the `SetSoftKeys` command, Flash Lite also supports the Left and Right soft keys.

This section contains the following topics:

Writing a key handler script 13
Creating a key catcher button (Flash Professional Only) 14
Creating a simple menu using movie clips (Flash Professional Only) 16

Writing a key handler script

To handle a keypress event, you attach an `on(keyPress "`*`key`*`")` handler to a button instance, where *`key`* is a supported key event name. For example, the following code, attached to a button instance on the Stage, executes when the user presses the Right Arrow key on their device:

```
on(keyPress "<Right>") {
  trace("You pressed the right arrow key");
}
```

The following table lists the ActionScript keypress event that Flash Lite generates in response to the user pressing a key on the device:

Device key	ActionScript key event	Availability
0-9, *, #	`0, 1, 2, 3, 4, 5, 6, 7, 8, -9, *, #`	All devices
Select key	`<Enter>`	All devices
Left Arrow key	`<Left>`	Devices that support four-way navigation, only.
Right Arrow key	`<Right>`	Devices that support four-way navigation, only.
Up Arrow key	`<Up>`	Devices that support four-way navigation, only.
Down Arrow key	`<Down>`	Devices that support four-way navigation, only.
Left soft key	`<PageUp>`	Devices that support the `SetSoftKeys` command, only.
Right soft key	`<PageDown>`	Devices that support the `SetSoftKeys` command, only.

Creating a key catcher button (Flash Professional Only)

If your application must handle several different keypress events you could either create a single button for each keypress event, or use a single button to handle all keypress events. This type of button is commonly called a *key catcher* (or key listener) button. Because the key catcher button isn't supposed to be visible to the user, it usually sits off the Stage (in the work area).

The following procedures demonstrate how to use a key catcher button to handle keypress events in a simple application. The application lets the user move a circle around the Stage by pressing the four arrow keys on their device.

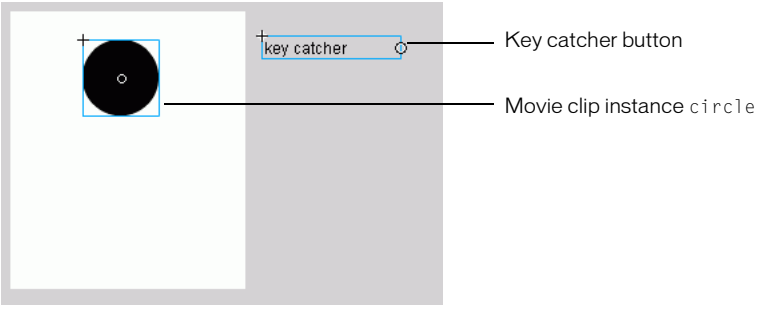
You can find a completed version of this application, key_catcher.fla, in the Samples folder on your hard disk:

- In Windows, browse to *boot drive*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\FlashLite.
- On the Macintosh, browse to *Macintosh HD*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/FlashLite.

To create and use a key catcher button:

1. Create a new document from the Flash Lite 1.1 Series 60 template, and save it as keycatcher.fla.

   For more information on creating documents from the Flash Lite templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.
2. Select the layer in the Timeline named Content.
3. Using the Oval tool, create a oval or circle on the Stage and convert it to a movie clip.
4. With the new movie clip selected, in the Property inspector, type **circle** in the Instance Name text box.
5. Using the Text tool, create a text field that contains the text **key catcher**, and convert it to a button symbol.
6. Position the new button symbol in the work area around the Stage. To view the work area around the Stage, select View > Work Area.



7. Select the key catcher button instance, and open the Actions panel (Window > Actions).
8. Enter the following code in the Actions panel:

```
on(keyPress "<Left>") {
  circle._x -= 10;
}
on(keyPress "<Right>") {
  circle._x += 10;
}
on(keyPress "<Up>") {
  circle._y -= 10;
}
on(keyPress "<Down>") {
  circle._y += 10;
}
```

9. Test the application by selecting Control > Test Movie.

   Press the four arrow keys on the emulator's keypad to make the circle move around the Stage.

For another example of using a key catcher button, see "Creating a simple menu using movie clips (Flash Professional Only)" on page 16.

Creating a simple menu using movie clips (Flash Professional Only)

In this section you'll learn how to create a simple menu using movie clips. In this method, rather than relying on the default tab navigation between buttons—and attaching code to each button—you use a key catcher button to listen for keypress events and update the user interface as needed. This technique does involve more development work than the button menu approach (see "Handling key events (Flash Professional Only)" on page 13), but it offers some advantages:

- Control over tab order. Rather than tab order being determined by the Flash Lite player (as with buttons), you (the developer) can decide what object has focus and how it responds to keypress events.
- Maintain menu selection between application states. For example, suppose you want your application to "remember" the last menu item the user selected so you can return the focus to the same item at a later time. This isn't possible using a button menu because you can't assign button focus using ActionScript.

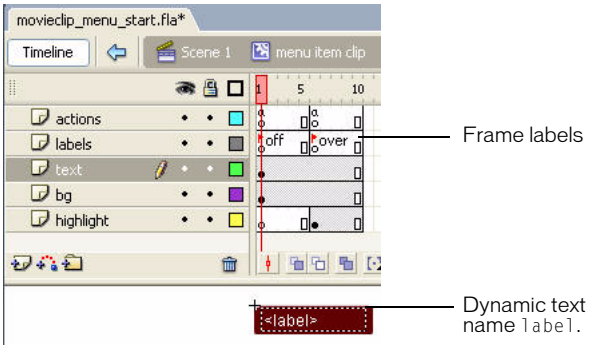
In the following procedure, you start with a partially completed Flash document. You can find a completed version of the application (movieclip_menu_complete.fla) in the /Tutorials and Samples/Samples/Flash Lite/ folder in the Flash installation folder.

To create a simple menu using movie clips:

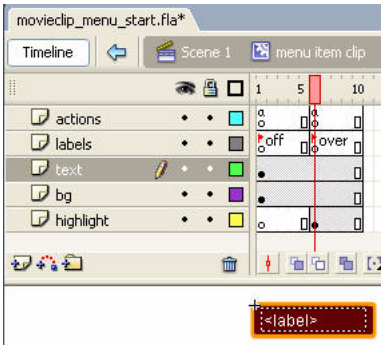
1. Open the file named movieclip_menu.fla located in the /Tutorials and Samples/Samples/Flash Lite/ folder in the Flash installation folder.
2. In the Timeline, select the layer named Menu Items.

3. Open the Library panel (Window > Library) and drag an instance of the movie clip symbol named Menu Item Clip from the Library panel to the Stage.

   This movie clip contains two keyframes, or visual states: one for the menu item's initial, unselected state, and the other for its selected state, which appears when the menu item receives focus. The following image shows the first keyframe of the movie clip's timeline. It contains a dynamic text field to display the menu item's label, and a red background graphic. The text field and background graphic extend across all frames in the movie clip's Timeline.

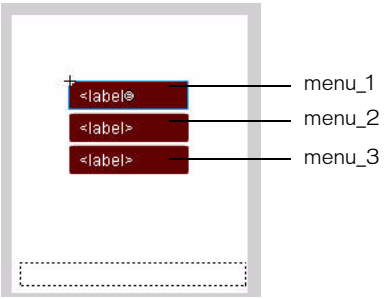


   The following image shows Frame 5 in the movie clip's Timeline. The only visual difference between this frame and the first one is the yellow border highlight that surrounds the menu item's red background.



4. Drag two more instances of the menu item movie clip to the Stage and align them vertically in a column.
5. Select the upper movie clip and, in the Property inspector, type **menu_1** in the Instance Name text box.

6. Assign the instance names of **menu_2** and **menu_3** to the middle and lower movie clips, respectively.



The numeric suffix appended to each instance name lets you dynamically refer to each movie clip in code, which you'll add shortly.

7. Using the Text tool, create a text field along the lower edge of the Stage.
8. In the Property inspector, select Dynamic from the Text Type pop-up menu, and type **status** in the Var text box.

   As in the simple menu example, this text field displays a status message about the menu item that is currently selected.
9. In the Timeline, select the first frame on the layer named Actions.
10. Open the Actions panel (Window > Actions), and enter the following code:

```
// Initialize menu item labels:
menu_1.label = "News";
menu_2.label = "Sports";
menu_3.label = "Weather";

// Initialize variable that specifies number of menu items
numItems = 3;

// Initialize selectedItem variable, which contains
// the index of the current menu selection
selectedItem = 1;

// Initialize status text field
currentLabel = eval("menu_" add selectedItem add ":label");
status = "Press to select " add currentLabel;

// Send the first menu item to its "over" state
tellTarget("menu_1") {
  gotoAndStop("over");
}
```

11. In the Timeline, select the layer named Key Catcher.

12. Open the Library and drag an instance of the button named key catcher to the Stage.

    Next you'll attach event handler code to this button that handles user keypress events and update the user interface.

13. With the button selected on the Stage, open the Actions panel.

14. Type (or copy and paste) the following code into the Actions panel:

```
on(keyPress "<Down>") {
  if(selectedItem < numItems) {
    // Turn off highlight on previously selected menu item:
    tellTarget("menu_" add selectedItem) {
      gotoAndStop("off");
    }
    // Increment selectedItem variable
    // and turn on highlight for new selection
    selectedItem++;
    tellTarget("menu_" add selectedItem) {
      gotoAndStop("over");
    }
    // Update status text field with label of selected item:
    currentLabel = eval("menu_" add selectedItem add ":label");
    status = "Press to select " add currentLabel;
  }
}

on(keyPress "<Up>") {
  if(selectedItem > 1) {
    // Turn off highlight on previously selected item:
    tellTarget("menu_" add selectedItem) {
      gotoAndStop("off");
    }
    // Increment selectedItem and turn on highlight for new selection
    selectedItem--;
    tellTarget("menu_" add selectedItem) {
      gotoAndStop("over");
    }
    // Update status text field with label of selected item:
    currentLabel = eval("menu_" add selectedItem add ":label");
    status = "Press to select " add currentLabel;

  }
}

on(keyPress "<Enter>") {
  // Update status field with selected item
  status = "You selected " add currentLabel;

}
```

15. Select Control > Test Movie to test the application in the emulator.

    To interact with the menu, click the Up and Down Arrow keys on the emulator with your mouse, or press the corresponding arrow keys on your keyboard.

Handling button events (Flash Professional Only)

Flash Lite supports the following ActionScript button events: `press`, `release`, `rollOver`, and `rollOut`. To handle these events, you attach an `on(event)` handler to a button instance, where *`event`* is one of the supported button events listed in the following table:

Button event	When event is generated
`press`	User presses the Select key on device when button has focus.
`release`	User releases the Select key on device when button has focus.
`rollOver`	Button receives focus.
`rollOut`	Button loses focus.

The following procedure demonstrates how to create a simple application that handles button events. For an example of using buttons to create a menu, see "Creating a simple menu using buttons and tab navigation (Flash Professional Only)" on page 22.

To create a button event handler script:

1. Create a new document from the Flash Lite 1.1 Series 60 device template and save it as button_handler.fla.

   For more information on creating documents from the Flash Lite templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.
2. Select Window > Common Libraries > Buttons to open an external library of prebuilt button symbols.
3. In the Library panel, double-click the Circle Buttons folder to open it.
4. Drag an instance of the Menu button symbol to the Stage.
5. Select the button and open the Actions panel (Window > Actions).

6. Type the following code in the Actions panel:

```
on(press) {
  trace("You pressed Button 1");
}
on(release) {
  trace("You released Button 1");
}
on(rollOver) {
  trace("Button 1 has focus");
}
on(rollOut) {
  trace("Button 1 lost focus");
}
```

7. Drag another instance of the same button to the Stage and position it directly below the first button.
8. With the second button selected on the Stage, open the Actions panel and enter the following code:

```
on(press) {
  trace("You pressed Button 2");
}
on(release) {
  trace("You released Button 2");
}
on(rollOver) {
  trace("Button 2 has focus");
}
on(rollOut) {
  trace("Button 2 lost focus");
}
```

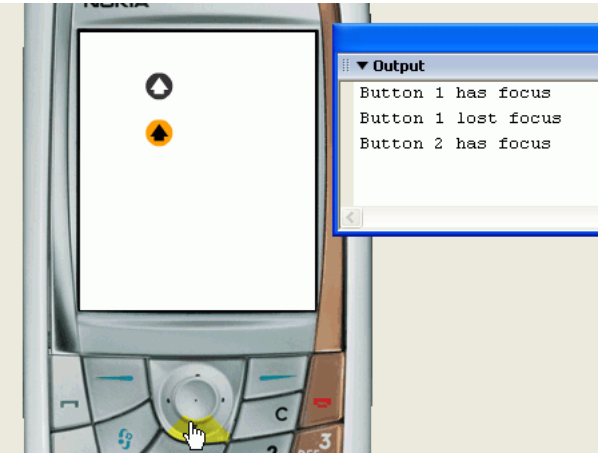
9. In the Timeline, select Frame 1 on the ActionScript layer.
10. Type the following code in the Actions panel:

```
_focusRect = false;
```

This disables the yellow focus rectangle that Flash Lite draws around the button with focus. In this case, the default focus rectangle is unnecessary because the button's contains an Over state that is displayed when it has focus.

11. Test the application in the emulator (Control > Test Movie).

    Watch the messages in the Output panel as you press the Up and Down Arrow keys on the emulator's keypad.



Creating a simple menu using buttons and tab navigation (Flash Professional Only)

This section shows you how to create a simple menu using buttons and tab navigation. To create the menu, you'll use three button symbols, one for each menu option. You'll attach event handling code to each button instance that displays a message when the user rolls over each menu item—that is, when the user gives focus to the corresponding button—and when the user selects the menu item by pressing the Select key on their device. For more information about handling button events in Flash Lite, see "Handling button events (Flash Professional Only)" on page 20.

You'll start with a partially completed Flash document that is preconfigured to target the Nokia 7610 and the Standalone Player content type. You can change these settings to target a different device and content type (see "Using the Flash Lite emulator (Flash Professional Only)" on page 66).

To create a simple menu using buttons:

1. Open the file named simple_menu_start.fla, which is located in the \Samples and Tutorials\Tutorial Assets\FlashLite\ folder.
2. Open the Library panel (Window > Library).

   Notice that the Library contains three buttons symbols named News Button, Weather Button, and Sports Button.

3. In the Timeline (Window > Timeline), select the layer named Menu Buttons.
4. Drag an instance of the News Button symbol from the Library panel to the Stage.
5. Repeat step 4 for the Sports and Weather buttons.
6. Align the three buttons vertically, as the following image shows:

7. In the Tools palette, select the Text tool and create a text field along the bottom of the Stage.

   This text field displays a short message when the user rolls over each menu item.
8. With the new text field still selected, do the following in the Property inspector:
   a. Select Dynamic Text from the Text Type pop-up menu.
   b. Type **status** in the Var text box.
9. On the Stage, select the News button and open the Actions panel (Window > Actions).

10. In the Actions panel, type the following code:

```
on(rollOver) {
  status = "Press to select News"
}
on(press) {
  status = "You selected news"
}
```

This code assigns some text to the dynamic text field when the user rolls over the News menu button.

11. Select the Sports button and type the following code in the Actions panel:

```
on(rollOver) {
  status = "Press to select Sports";
}
on(press) {
  status = "You selected Sports";
}
```

12. Select the Weather button and type the following code in the Actions panel:

```
on(rollOver) {
  status = "Press to select Weather";
}
on(press) {
  status = "You selected Weather";
}
```

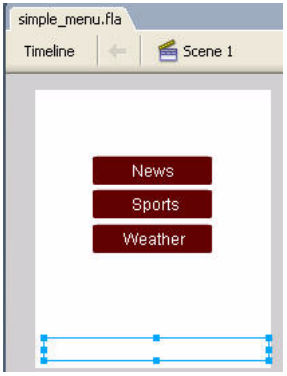
13. In the Timeline, select Frame 1 of the layer named Actions.

14. In the Actions panel, type the following code:

```
_focusRect = false;
```

This disables the yellow focus rectangle that Flash Lite draws by default around buttons and text fields that have focus (see "About the focus rectangle" on page 11).

At this point, the Stage should look something like the following image:



15. Select Control > Test Movie to preview the application in the emulator.

Click the Down Arrow key on the emulator with your mouse (or press the Down Arrow key on your computer's keyboard) to navigate between menu options; to select a menu item, click the emulator's Select key by using your mouse (or press the Enter key on your computer's keyboard).

Using the soft keys (Flash Professional Only)

A device's soft keys are multifunctional keys that use the device's display to identify their purpose at any moment. For example, in the following application, labels above the soft keys indicate that the user can press the Right soft key to view the next dinner special, or press the Left soft key to return to the application's home screen:

To use the Left and Right soft keys, you must first call the `SetSoftKeys` command (see `SetSoftKeys` in *Flash Lite 1.x ActionScript Language Reference*). Subsequently, Flash Lite generates a `PageDown` event when the user presses the Right soft key and a `PageUp` event when the user presses the Left soft key. You write ActionScript event handler code that responds to these events and takes the desired action.

The `SetSoftKeys` command takes two parameters that specify the labels for the Left and Right soft keys, respectively, that appear when your application is **not** running in full-screen mode. For applications running in full-screen mode, the labels that you specify are not visible, so you must create your own labels and position them on the Stage where the soft keys are located.

For example, consider the following `SetSoftKeys` command call:

```
fscommand2("SetSoftKeys", "Options", "Exit");
```

The following image shows the result of this command on an application running on an actual device in normal (not full-screen) mode:

If you enable full-screen mode—that is, if you call `fscommand("fullscreen", true)`—the labels that you specify as parameters to the `SetSoftKeys` command are not visible. Consequently, in full-screen mode applications, you must create your own soft key labels, as the following image shows:

To use the soft keys in an application:

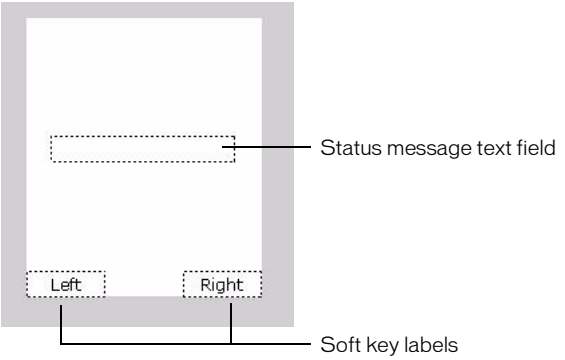
1. Create a new document from the Flash Lite 1.1 Series 60 device template.

   For more information about device templates, see "Using Flash Lite document templates (Flash Professional Only)" on page 16.

2. Open the Device Settings dialog box, and select the Standalone Player content type. Add one of the devices in the T-Mobile > Nokia folder to your list of test devices.

3. In the Timeline, select the Content layer.
4. Using the Text tool, create a static text field named Left (or text of your choice) and position it in the lower-left corner of the Stage, above the Left soft key on the device.
5. Create another static text field named Right, and position it in the lower-right corner of the Stage, above the Right soft key on the device.
6. Using the Text tool, create another dynamic text field and position it in the middle of the Stage.

   This text field displays a message when you run the application and press the Left and Right soft keys. Your document's Stage should look like the following image:



NOTE

In a real-world application, you might want to use something other than ordinary text fields for the soft key labels, such as graphic or movie clip symbols.

7. With the status text field still selected, in the Property inspector, type **status** in the Var text box.
8. Create a key catcher button (see "Creating a key catcher button (Flash Professional Only)" on page 14). In the Actions panel, attach the following code to the button:

```
// Handle Left soft keypress event
on(keyPress "<PageUp>") {
  status = "You pressed the Left soft key.";
}
// Handle Right soft keypress event
on(keyPress "<PageDown>") {
  status = "You pressed the Right soft key.";
}
```

9. In the Timeline, select Frame 1 on the Actions layer.

10. In the Actions panel, type the following code:

```
fscommand2("SetSoftKeys", "Left", "Right");
fscommand2("FullScreen", true);
```

The two parameters of the `SetSoftKeys` command—Left and Right, in this case—specify the labels that Flash Lite displays above the soft keys when the application is not being viewed in full-screen mode. In this case, the application uses the `FullScreen` command (see `FullScreen` in *Flash Lite 1.x ActionScript Language Reference*) to force the application to display in full-screen mode. Consequently, the values you choose for those parameters can be arbitrary text strings or expressions.

```
fscommand2("SetSoftKeys", foo, bar);
```

11. Select Control > Test Movie to test the application in the emulator.

Click the Left and Right soft keys on the emulator with your mouse, or press the Page Up and Page Down keys on your keyboard to test the application.

CHAPTER 2

Working with Text and Fonts

2

This chapter describes how you can add static and dynamic text fields, and add input text fields to your Macromedia Flash Lite applications.

This chapter contains the following topics:

About text in Flash Lite 31
Using input text fields 34
Font rendering methods in Flash Lite 38
Flash Lite rendering quality and anti-aliased text 39
Embedding font outlines in SWF files 40
Text field example application (Flash Professional Only) 42
Creating scrolling text (Flash Professional Only). 45

About text in Flash Lite

Flash Lite supports three types of text fields: static, dynamic, and input. A static text field has content that doesn't change during playback. For example, you might use static text fields to display page titles or labels. For more information about creating a static text field, see "Text field example application (Flash Professional Only)" on page 42.

Dynamic text fields let you control the content at runtime. You can associate an ActionScript variable name with a dynamic text field that you can refer to in code. For example, if you are making a calculator in Flash Lite, you would use a dynamic text field to show calculation results. For more information about using dynamic text fields, see "Assigning a variable name to a text field" on page 32.

An input text field is like a dynamic text field, except that a user can interact with an input text field to open the device's generic input dialog box to provide text input. For more information about using input text fields, see "Using input text fields" on page 34.

About font rendering methods in Flash Lite

To render text on a device's display, Flash Lite can either use fonts that are available on the device or use font data that is embedded in the SWF file. Device fonts have the advantage of smaller SWF file sizes but give less control over the font display. When you embed font data in the SWF file, you have more control over the font display, but it increases the file size.

For embedded font data, Flash Lite supports both anti-aliased and aliased (bitmap) text, which makes text more readable at small point sizes. You can also use pixel fonts from third-party font designers to make small text readable. For more information, see "Font rendering methods in Flash Lite" on page 38 and "Embedding font outlines in SWF files" on page 40.

Text features in Flash Player not supported in Flash Lite 1.x

Text fields in Flash Lite do not support the following features available in desktop versions of Flash Player:

- The enhanced font rendering technology available in Flash Player 8 and later.
- HTML, CSS, or other rich text formatting features.
- The TextField and TextFormat objects.

Assigning a variable name to a text field

You can associate dynamic and input text fields with an ActionScript variable name, which lets you get or set the contents of the text using ActionScript.

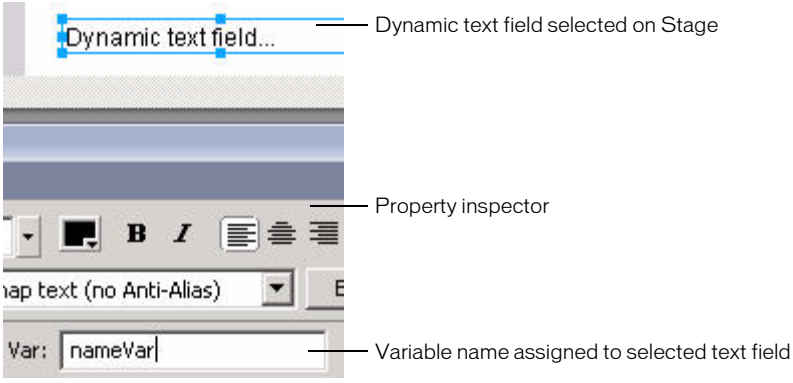
To associate a variable name with a dynamic or input text field:

1. Create a new document from the Flash Lite 1.1 Symbian Series 60 template and save it as dynamic_text.fla.

   For more information about using Flash Lite document templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.
2. Create a dynamic or input text field on the Stage, and select it.

3. In the Property inspector, in the Var text box, type **nameVar**.

   The value that you enter must be a valid variable identifier—the first character must be a letter, underscore (_), or dollar sign ($), and each subsequent character must be a letter, number, underscore, or dollar sign.



4. In the Timeline, select Frame 1 on the layer named ActionScript.
5. Open the Actions panel (Window > Actions) and enter the following code:

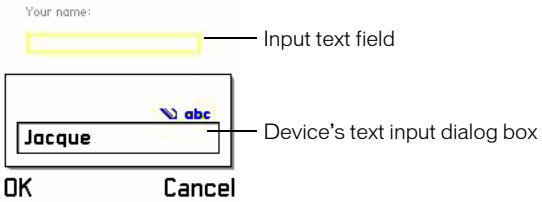
   ```
   nameVar = "George Washington"
   ```

6. Test the application in the emulator (Control > Test Movie).

For another example of using dynamic text fields, see "Text field example application (Flash Professional Only)" on page 42.

Using input text fields

Input text fields in Flash Lite, like dynamic text fields, let you get and set their contents at runtime with ActionScript. In addition, input text fields let Flash Lite applications get user input using the device's generic input text dialog box. (Flash Lite does not support inline text input.) The following image shows an input dialog box on a Symbian Series 60 device:



To open the device's input dialog box, the user must first give an input text field focus, and then press their device's Select key. By default, Flash Lite draws a yellow rectangle around the input text field that has focus.

The text input dialog box is modal, which means that the user can't interact with the Flash content while the dialog box has focus. Flash Lite also pauses the playhead in the Flash application while the dialog box has focus.

If the user clicks OK (the Left soft key), the text input dialog box closes, and Flash Lite automatically assigns the text to the input text field. If the user clicks Cancel (the Right soft key), no text is assigned to the input text field.

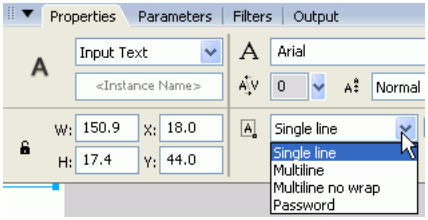
The Flash Lite emulator mimics the features of the text input dialog box when you test your application in the Flash authoring tool. The following image shows the text input dialog running in the emulator:

Text input dialog running in emulator

For an example of using an input text field in an application, see "Text field example application (Flash Professional Only)" on page 42.

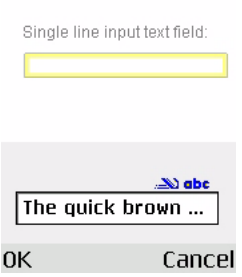
Types of input text fields

Flash Lite supports single line, multiline, and password input text fields. You specify an input text field's type by using the Line Type pop-up menu in the Property inspector, as the following image shows:

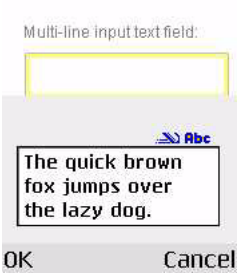


The line type that you specify for an input text field determines the behavior of the device's input text dialog box when a user edits the text field.

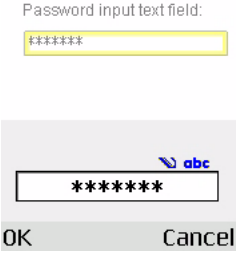
For example, when a user edits a single line input text field, the device's input text dialog box displays a single line input text box. The input text box scrolls horizontally if the user enters more characters than can display. The following image shows a device's input text dialog box for a single line input text field:



When a user edits a multiline input text field, the device's input text dialog box expands as necessary to display all the text the user enters, as the following image shows:



When a user edits a password input text field, the device's input text dialog box masks each character that the user enters with an asterisk (after a short delay).



Restricting character input

You can use the `SetInputTextType` command to restrict the characters that the user can enter in the text input dialog box. For example, suppose an application contains an input text field for the user to provide a numeric value, such as their age. And furthermore suppose that the input text field has the variable name of `ageVar`.

To ensure that the user only enters numeric values into the text input dialog, you could add the following code to your application:

```
fscommand2("SetInputTextType", "ageVar", "Numeric");
```

When the user opens the input text dialog box, they will only be able to enter numeric values in the text fields.

For more information, see `SetInputTextType` in *Flash Lite 1.x ActionScript Language Reference*.

Input text fields and the focus rectangle

By default, Flash Lite draws a yellow rectangle around the input text field that has keypad focus, as the following image shows:

You can disable the focus rectangle by setting the global `_focusRect` property to `false`. However, in that case, the user won't be able to see that the text field has keypad focus and won't know to press the Select key on their device. Macromedia recommends that you do not disable the focus rectangle when using input text fields.

For more information about controlling the behavior of the yellow focus rectangle, see "About the focus rectangle" on page 11 and `_focusrect` in *Flash Lite 1.x ActionScript Language Reference*.

Font rendering methods in Flash Lite

Flash Lite can render text field fonts in any of the following ways:

Use fonts that are available on the device You can apply a font to a text field that you know is available on the device, or you specify one of the three generic device fonts (_sans, _serif, or _typewriter) that are available in the Font pop-up menu. If you select a generic device font, Flash Lite tries to match the selected generic font to a font on the device (for example, _sans is mapped to a sans serif font, if available) at runtime.

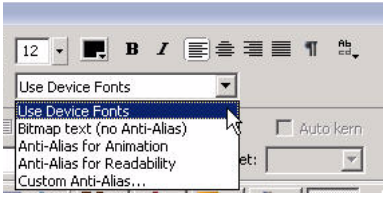
Render the font as a bitmap Flash Lite renders bitmap text by aligning font outlines to pixel boundaries, which makes text readable at small point sizes (such as 10 points or smaller). This option requires that you include font outlines in the published SWF file for the selected font (for more information, see "Embedding font outlines in SWF files" on page 40).

Render the font as anti-aliased vectors Flash Lite renders anti-aliased text using vector-based representations of font outlines, which you embed in the published SWF file (for more information, see "Embedding font outlines in SWF files" on page 40).

You select a font rendering method for a text field using the Font Rendering Method pop-up menu located in the Property inspector. The Font Rendering Method pop-up menu contains five rendering options; however, only three of those are available to Flash Lite 1.0 and 1.1 developers. The other two methods (Anti-Alias for Readability and Custom Anti-Alias) are only available to applications that are targeting Flash Player 8 or later.

To select a font rendering method for a text field:

1. Select a text field on the Stage.
2. In the Property inspector, select one of the following options from the Font Rendering Method pop-up menu:



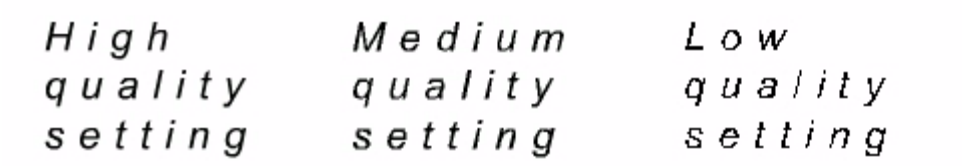
   - Select Use Device Fonts to have Flash Lite use a font that is available on the device. No font data is embedded in the published SWF file.
   - Select Bitmap text (no Anti-Alias) to have Flash Lite align font outlines along pixel boundaries, which makes small text appear crisp and clear. This option requires that Flash embed font outlines in the published SWF file (for more information, see "Embedding font outlines in SWF files" on page 40).
   - Select Anti-Alias for Animation to have Flash Lite anti-alias the text field's font according to the current rendering quality setting (for more information, see "Flash Lite rendering quality and anti-aliased text" on page 39). This option requires that Flash embed font outlines in the published SWF file (for more information, see "Flash Lite rendering quality and anti-aliased text" on page 39).

Flash Lite 1.0 and 1.1 do not support the Anti-Alias for Readability or Custom Anti-Alias font rendering options. Those rendering options are available only in Flash Player 8 and later on desktop computers.

Flash Lite rendering quality and anti-aliased text

Flash Lite has three rendering quality settings: low, medium, and high. The higher the rendering quality setting, the more smoothly and accurately Flash Lite renders vector outlines; a lower quality setting results in less smoothly drawn outlines. By default, Flash Lite rendering outlines using medium quality. You can control the rendering quality using the `SetQuality` command (for more information, see `SetQuality` in *Flash Lite 1.x ActionScript Language Reference*).

Flash Lite renders anti-aliased text using vector representations of font outlines. If you want anti-aliased text to appear as smooth as possible, you should set the player's rendering quality to high. Rendering quality affects all vector content on the screen, not just anti-aliased text. The following figures show an anti-aliased text field (Arial, 24 point) rendered at high, medium, and low qualities:



To maximize animation performance and frame rate—for example, during an intensive animation or tween sequence—you can temporarily set the rendering quality to a lower setting and return it to the previous setting after the animation has completed.

Embedding font outlines in SWF files

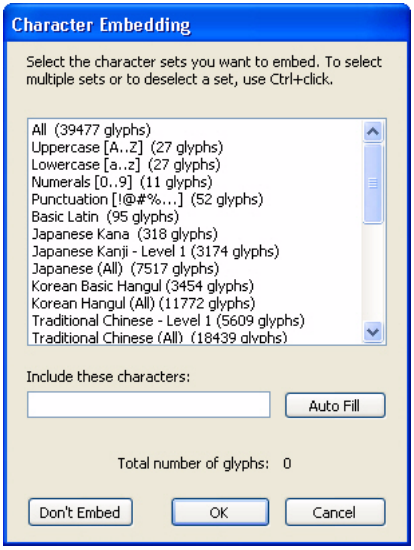
To render a text field's font, Flash Lite can either use fonts that are available on the device or use font outlines that are embedded in the published SWF file (for more information, see "Font rendering methods in Flash Lite" on page 38). Embedding font outlines in the SWF file ensures that the text field's font appears the same on all target platforms, but results in larger file size. Flash Lite requires font outlines to render either bitmap (no anti-alias) or anti-aliased text.

For static text fields that use the anti-alias or bitmap font rendering methods, Flash automatically embeds the font outlines required to display the text field's contents. For example, if a static text field contains the word *Submit*, Flash automatically embeds the font outlines required to display those six characters (that is, *S*, *u*, *b*, *m*, *i*, and *t*). Because the contents of a static text field can't change, the SWF file need only include fonts outlines for those specific characters.

For dynamic and input text fields that use the anti-alias or bitmap font rendering methods, you must specify the characters whose font outlines you want to embed in the published SWF file. The contents of those types of text fields can change during playback; consequently, Flash can't assume which font outlines need to be available. You can include font outlines for all characters in the selected font, a range of characters, or specific characters. You use the Character Embedding dialog box to specify which characters you want to embed in the published SWF file.

To embed font outlines for a dynamic and input text field:

1. Select the dynamic or input text field on the Stage.
2. In the Property inspector, select Bitmap (no anti-alias) or Anti-Alias for Animation from the Font rendering method pop-up menu.
3. Click the Embed button located next to the Font rendering method menu to open the Character Embedding dialog box.



4. Select the characters that you want to embed from the list, type the specific characters that you want to embed in the text box, or click Auto Fill to include the characters present in the selected text field.
5. Click OK.

Text field example application (Flash Professional Only)

This section describes how to create a simple application that uses static, dynamic, and input text fields. The application asks the user (using text in a static text field) to provide their name (using an input text field). After they enter their name and select a button, the application shows a message (using a dynamic text field) addressed to the user.

You can find a completed sample file for this section, textfield_example.fla, in the Samples folder on your hard disk.

- In Windows, browse to *boot drive*\Program Files\Macromedia\Flash 8\Samples and Tutorials\Samples\FlashLite.
- On the Macintosh, browse to *Macintosh HD*/Applications/Macromedia Flash 8/Samples and Tutorials/Samples/FlashLite.

To create a text field example application:

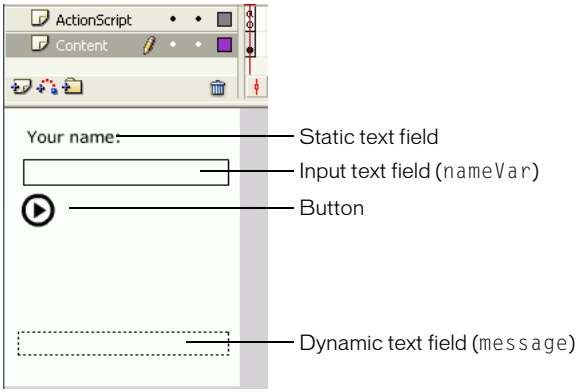
1. In Flash, create a new document from the Flash Lite 1-1 - Symbian Series 60 device template.

   For more information about creating documents from device templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.
2. In the Timeline (Window > Timeline), select the first frame of the layer named ActionScript.
3. Open the Actions panel (Window > Actions).
4. Type `stop();` in the Actions panel to stop the playhead on that frame.
5. Select the Text tool in the Tools panel to create a text field on the Stage that contains the text "Your name:".
6. With the new text field selected, in the Property inspector, select Use Device Fonts from the Font Rendering Method pop-up menu.
7. Create a new text field below the first one and, in the Property inspector, select Input Text from the text type pop-up menu, select Use Device Fonts from the Font Rendering Method pop-up menu, type **nameVar** in the Var text box, and select the Show Border Around Text option.

8. Select Window > Common Libraries > Buttons to open a library of prebuilt button symbols.
9. In the Buttons library, double-click the Circle Buttons folder to open it.
10. Drag an instance of the symbol named "circle button - next" to the Stage.
11. Using the Text tool create another text field at the bottom of the Stage.

    This text field will display a message that contains the name the user enters in the input text field.
12. With the new text field selected, in the Property inspector, type **message** in the Var text box, select Dynamic from the Text Type pop-up menu, and select Use Device Fonts from the Font Rendering Method pop-up menu.

    The Stage in your application should look similar to the following image:



13. Select the button on the Stage, and open the Actions panel (Window > Actions).
14. Enter the following code in the Actions panel:

```
on(press) {
  message = "Hello, " add nameVar;
}
```

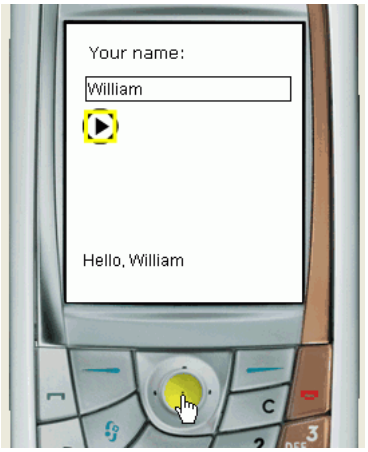
15. Select Control > Test Movie to test the application in the emulator.

a. Press the Down arrow on the emulator's keypad to give the input text field focus.

b. Press the Select key on the emulator to open the emulator's text input dialog box, and type your name using your computer's keyboard.

c. Press OK to close the input text dialog box.

d. Press the Down Arrow key on the emulator's keypad again to give focus to the button, and press the Select key.



Creating scrolling text (Flash Professional Only)

Flash Lite 1.1 supports the `scroll` and `maxscroll` text field properties, which let you create scrolling text fields. The `scroll` property specifies the first visible line in a text block; you can get and set its value. For example, the following code scrolls the text field whose variable name is `story_text` down by five lines:

```
story_text.scroll += 5;
```

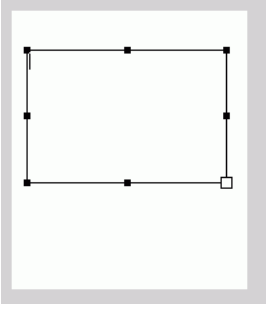
The `maxscroll` property specifies the first visible line in a text block when the last line of the text is visible in the text block; this property is read-only. You can compare a text field's `maxscroll` property to its `scroll` property to determine how far a user has scrolled within a text field. This is useful if you want to create a scroll bar that provides feedback about the user's current scroll position relative to the maximum scroll position.

To create a scrolling text field and control it with ActionScript:

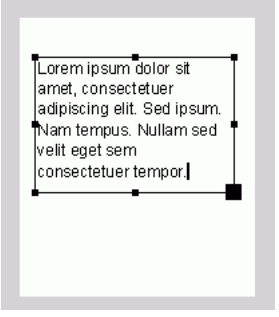
1. In Flash, create a new document from the Flash Lite 1-1 - Symbian Series 60 device template.

   For more information about creating documents from device templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.

2. Using the Text tool, click the Stage, and then drag a text field approximately the size shown in the following image on the Stage:



3. Select Multiline from the Line type pop-up menu in the Property inspector.
4. Select Dynamic Text from the Text Type pop-up menu in the Property inspector.
5. Select Use Device Fonts from the Font rendering method pop-up menu in the Property inspector.
6. Select Text > Scrollable to make the text field scrollable.
7. Type **story** in the Var text box in the Property inspector. This associates the ActionScript variable named `story` with the text field.
8. Double-click inside the text field, and enter enough text so that one or more lines of text extend below its lower edge.



9. Create a new button symbol, and add an instance of it to the Stage or in the area off the Stage.

   This button acts as *key catcher* button, and it doesn't need to be visible to the user. For more information about creating key catcher buttons, see "Creating a key catcher button (Flash Professional Only)" on page 14.

10. Select the button, and open the Actions panel (Window > Actions).

11. Enter the following code in the Actions panel:

```
on(keyPress "<Down>") {
  story.scroll++;
}
on(keyPress "<Up>") {
  story.scroll--;
}
```

12. Select Control > Test Movie to test the application in the Flash Lite emulator.

    Press the Up and Down Arrow keys on your keyboard (or the Up and Down buttons on the emulator's keypad) to scroll the text up or down.

For simplicity, this example lets you enter the text field's content in the authoring tool. But you can easily modify the example so that the text field's content is updated using ActionScript. To do this, you would write ActionScript that assigns the desired text to the variable name that you assigned to the multiline text field (`story`, in this example).

```
story = "Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed
  ipsum. Nam tempus. Nullam sed velit eget sem consectetuer tempor. Morbi
  eleifend venenatis pede. Cras ac lorem eget massa tincidunt
  iaculis...etc."
```

CHAPTER 3

Working with Sound

3

Macromedia Flash Lite 1.1 supports device sounds and standard, or native, Flash sound. This chapter describes what you must do to incorporate sound with your applications on mobile devices.

This chapter contains the following topics:

About sound in Flash Lite 49
Using device sound (Flash Professional Only) 50
Using native Flash sounds 56

About sound in Flash Lite

Flash Lite 1.1 supports two types of sound: device sound and standard (*native*) Flash sound. Device sounds are stored in the published SWF file in the device's native audio format, such as MIDI or MFi. To play the device sound, Flash Lite passes the sound data to the device, which then decodes and plays the sound. Flash Lite cannot synchronize device sounds with animation. Flash Lite 1.0 supports only device sounds.

In addition to the device audio formats that are supported by Flash Lite 1.0, Flash Lite 1.1 supports native Flash audio—the same type of sound supported by the desktop version of Macromedia Flash Player. Unlike device sounds, you can synchronize native sounds to animation on the Timeline.

Flash Lite 1.0 supports MIDI and MFi device audio formats. In addition to these formats, Flash Lite 1.1 supports the Synthetic music Mobile Application Format (SMAF) device audio format, as well as native Flash sound compressed with PCM (or WAV), ADPCM, or MP3 audio compression.

Event and stream (synchronized) sound

Flash Lite 1.1 supports event and stream (*synchronized*) sound. Event sounds play independently of the Timeline and continue to play until either the end of the sound buffer has been reached, or the sound is stopped using ActionScript. Event sounds must download completely before they begin playing.

Stream sounds are synchronized with the Timeline on which they reside and are often used to synchronize audio with animation. Stream sounds stop when the playhead of the containing Timeline is stopped. During playback, Flash Lite drops frames from the animation, if required, to keep the sound playback synchronized with animation.

Only native Flash sound can be synchronized with the Timeline; you can use device sounds only as event sounds. Flash Lite 1.0 supports only event sound.

Using device sound (Flash Professional Only)

A *device sound* is a sound that is encoded in the device's native audio format, such as MIDI or MFi. The Flash authoring tool does not let you directly import device sound files into a Flash document; rather, you first import a proxy sound in a supported format such as MP3, WAV, or AIFF. You then link the proxy sound to an external mobile device sound, such as a MIDI file. During the document publishing process, the proxy sound is replaced with the linked external sound. The SWF file generated contains the external sound and uses it for playback on a mobile device.

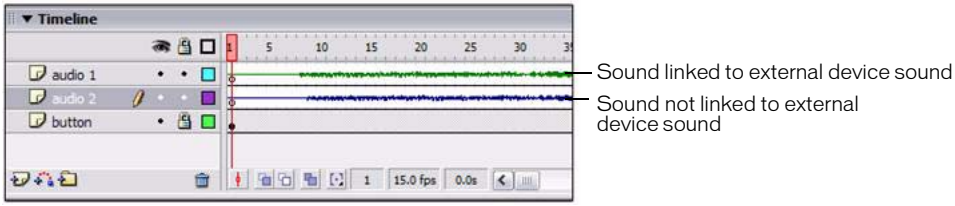
When using device sounds in Flash Lite, keep the following constraints in mind:

- Device sounds can only be used as event sounds; you can't synchronize device sounds to the Timeline.
- Flash Lite does not support the Effect, Sync, and Edit options for device sounds.
- You must specify an external device sound file for each sound in a document.
- As with all external files, the device sound file or the sound bundle file must be available when you publish your SWF file, but is not needed by the SWF file for playback.

You can also bundle multiple device sounds together in a single file. This is useful if you're creating the same content for several devices that support different device sound formats. For more information, see "Using compound sound (Flash Professional Only)" on page 54.

In Flash Lite 1.1, a device sound can play at any time. In Flash Lite 1.0, a device sound can only play in response to a user pressing a key on their device. For more information, see "Triggering device sounds in Flash Lite 1.0" on page 53.

The Timeline in the Flash authoring tool displays sound waveforms, as the following image shows. Waveforms for sounds that are linked to external device sounds are colored green; waveforms for sounds that are not linked to external device sounds are colored blue.



This section contains the following topics:

Adding a device sound to a button (Flash Professional Only) 51
Triggering device sounds in Flash Lite 1.0 53
Using compound sound (Flash Professional Only) 54

Adding a device sound to a button (Flash Professional Only)

The following procedure describes how to add a device sound to a button symbol's timeline so that the sound plays then user "clicks" the button (that is, when the user presses the selects on their device when the button has focus). To do this, you attach the proxy sound to the Down frame in a button symbol's timeline. You then associate the device sound that you want to play with the proxy sound.

A completed version of the application named button_sound_complete.fla is located in the *flash_install_dir*/Tutorials and Samples/Samples/Flash Lite/Sound/ folder.

To add a device sound to a Flash document:

1. Create a new document from the Flash Lite 1.0 Symbian Series 60 document template, and save it as device_sound.fla.

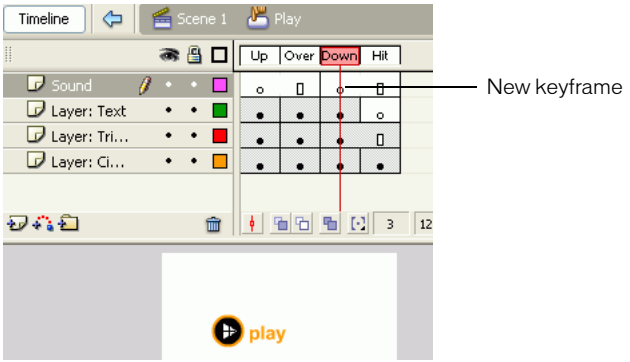
   For more information about using the Flash Lite document templates, see "Using Flash Lite document templates (Flash Professional Only)" in *Getting Started with Flash Lite*.
2. Select File > Import > Import to Library. Locate the *flash_install_dir*/Tutorials and Samples/Samples/Flash Lite/Sound/ folder, and select the proxy.wav file. Click OK.

   This sound file acts as the proxy sound for the device sound that you want to include.
3. Select Window > Common Libraries > Buttons.

   This opens an external library that contains prebuilt button symbols.
4. In the Buttons library, double-click the Circle Buttons folder to open it.

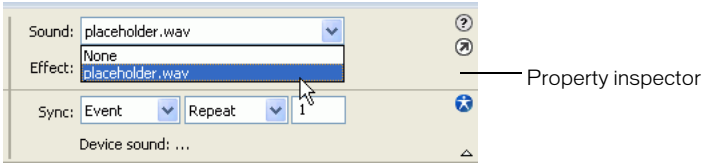
5. In the Timeline, select the layer named Content in the Timeline.
6. Drag the button symbol named Play from the buttons library to the Stage.
7. Double-click the new button to open it edit mode.

   The Timeline changes to show the button's frames named Up, Over, Down, and Hit.
8. Select Insert > Timeline > Layer to create a new layer.
9. Select Modify > Timeline > Layer Properties, and change the name of the layer to Sound.
10. Select the Down frame in the Sound layer and press the F6 function key to insert a new keyframe.



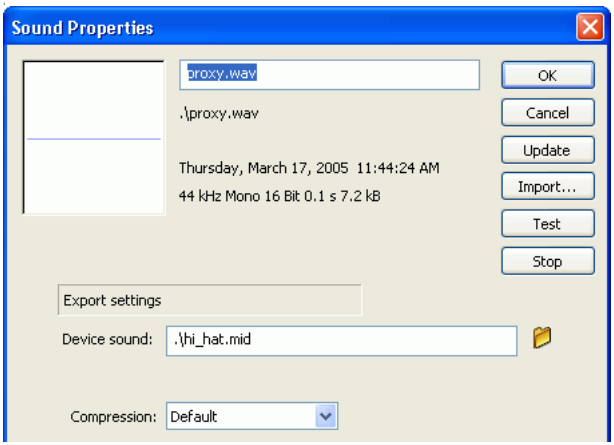
11. With the new keyframe selected in the Timeline, in the Property inspector, select proxy.wav from the Sound pop-up menu.

    This attaches the proxy sound to the keyframe.



12. To link the proxy sound with the actual device sound file, do the following:
    a. In the Library, double-click the proxy.wav symbol to open the Sound Properties dialog box.
    b. In the Sound Properties dialog box, click the folder icon to the right of the Device sound text box to open the Select Device Sound dialog box.

c. Browse to the *flash_install_dir*/Tutorials and Samples/Samples/Flash Lite/Sound/ and select the file named hi_hat.mid.



d. Click OK.

13. Select Control > Test Movie to start the Flash Lite emulator and test your SWF file.

    In the emulator, press the Down Arrow key on the keypad to give the Play button focus, and then press the Select key to play the sound.

Triggering device sounds in Flash Lite 1.0

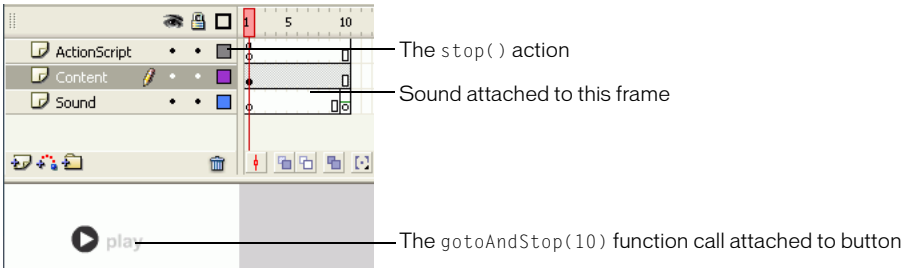
In Flash Lite 1.0, a device sound can only play in response to a user pressing a key on their device. There are two ways to satisfy this constraint. One way is to attach the sound to the Down frame in a button symbol's timeline. When the button has focus, and the user presses the Select key on their device, Flash Lite plays the sound in the button's Down frame. For an example of this technique, see "Adding a device sound to a button (Flash Professional Only)" on page 51.

The other way to trigger a device sound in Flash Lite 1.0 is to have the user press a key that sends the playhead to a frame in the timeline that contains a device sound. Any device sound at that frame plays when the playhead enters the frame.

For example, suppose your application has a button on the Stage on Frame 1, and a device sound attached to Frame 10. You attach the following `on(press)` handler to the button instance:

```
on(press) {
  gotoAndStop(10);
}
```

The following image shows how the application might look in Flash:



When the user select the button, the sound on Frame 10 plays. This technique requires that the device sound be attached to the same frame specified in the `gotoAndPlay()` function. For instance, in the example discussed above, if the sound were attached to Frame 11 rather than Frame 10, Flash Lite would not play the sound when the playback head reached Frame 11.

Using compound sound (Flash Professional Only)

Flash Lite 1.1 provides the ability to encapsulate device-specific sounds of multiple formats into a single tagged data block. This provides content developers with the ability to create a single piece of content that is compatible with multiple devices. As an example, a single Flash application can contain the same sound represented in both MIDI and MFi formats. You can play this Flash application both on a device that supports only MIDI and on a device that supports only MFi, with each device playing back the specific sound format that it natively supports.

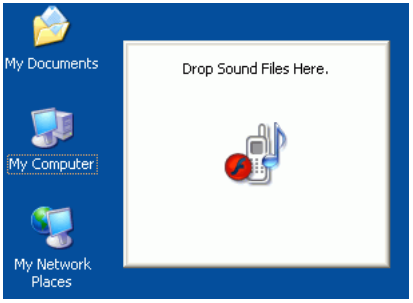
You use a utility called the Flash Lite Sound Bundler to bundle multiple device sounds into a single sound bundle (FLS) file. You then add the FLS file to your Flash Lite document the same way you add a standard device sound, by first importing a proxy sound into your Flash document, and specifying the sound bundle file to replace the proxy sound at publish time. For more information about adding device sounds to your Flash Lite applications, see "Adding a device sound to a button (Flash Professional Only)" on page 51.

NOTE	As of this writing, the Sound Bundler utility is only supported by Windows systems.

To create a sound bundle file:

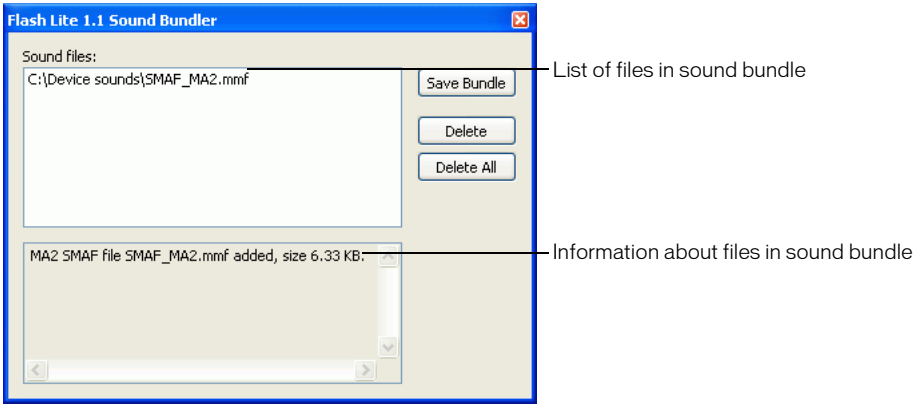
1. Open the Flash Lite Sound Bundler application (FlashLiteBundler.exe) located in the Flash 8 install folder (for example, *boot drive*/Program Files/ Macromedia/Flash 8/ FlashLiteBundler.exe).

   The Sound Bundler appears as a floating window.



2. From your desktop, drag the first sound file to be bundled into the floating window.

   The Flash Lite 1.1 Sound Bundler window appears. The upper part of the Sound Bundler window is a list of the files you added to the sound bundle. The lower part of the window contains information about the sounds in the sound bundle, including sound format, size of sound data, and filename.

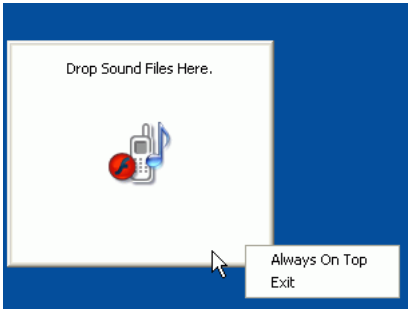


3. Drag the rest of the sound files that you want to bundle into the window.

   You can't bundle more than one file in a given audio format. For example, you can't bundle two MIDI files in the same FLS file.

4. To delete a file from the sound bundle, select in the list of sound files and click Delete. To delete all files in the sound bundle, click Delete All.

5. Click Save Bundle to save the FLS file.
6. To exit from the Sound Bundler, right-click on the Sound Bundler window and select Exit.



The next step is to add the sound bundle (FLS) file to your Flash document. The process is the same as adding standard device sounds to Flash documents, except that instead of specifying a single device sound file to replace the proxy sound, you specify the FLS file that you created. For more information, see "Using device sound (Flash Professional Only)" on page 50).

Using native Flash sounds

Flash Lite 1.1 supports standard, or *native*, Flash sounds, in addition to the device sound formats supported by Flash Lite 1.0. Native sound in Flash Lite 1.1 can play either as event sound or synchronized sound (see "Event and stream (synchronized) sound" on page 50).

The general workflow for adding native sounds to your Flash Lite 1.1 content is the same as for Flash desktop applications, with the following exceptions:

- Flash Lite does not support loading external MP3 files.
- Flash Lite does not support the Sound object.
- Flash Lite does not support the Speech audio compression option (see "Compressing sounds for export" in *Using Flash*).

For more information about working with native sound in Flash, see the following topics in *Using Flash*:

- "Importing sounds" in *Using Flash*.
- "Adding sounds to a document" in *Using Flash*.
- "Starting and stopping sounds at keyframes" in *Using Flash*.
- "Compressing sounds for export" in *Using Flash*.

CHAPTER 4

Optimizing content for performance and file size

4

This chapter provides tips and techniques for optimizing your Flash Lite for file size and performance.

This chapter contains the following topics:

SWF file size and memory 57
Performance optimization 57
Device speed and frames per second 61

SWF file size and memory

The runtime memory available to Flash Lite applications running on mobile phones is limited and might vary among models. Generally, for mobile phones, this limit is not less than 1 MB. Because Macromedia Professional 8 does not provide a way to check a phone's runtime memory consumption, Macromedia strongly recommends that you test all content on actual mobile phones.

Performance optimization

The CPU speed in mobile phones varies among models and is typically much slower than the CPU speed in current desktop computers. Therefore, it is important to consider application performance and optimization from the beginning of each project for creating Flash Lite content created for mobile phones.

This section contains the following topics:

Animation guidelines 58
Bitmap graphics 58
Optimizing ActionScript 61

Animation guidelines

When creating animated content for a mobile phone, it is important to consider the phone's CPU limitations. The following guidelines can help prevent your Flash Lite content from running slowly:

- Flash Lite can render vector graphics in your application at three different quality levels: low, medium, and high. The higher the rendering quality, the more smoothly and accurately that Flash Lite renders vector graphics, and the more processing that is required of the device's CPU. If you need to provide complex animation, experiment with changing the player's quality setting of the content. Because changing the quality setting can noticeably affect the visual quality of the Flash Lite content, you should thoroughly test the SWF file.

  To control the rendering quality of a SWF file, you can use the `_quality` property or the `SetQuality` command.

  For the `_quality` property, valid values are `LOW`, `MEDIUM`, and `HIGH`. The following code sets the rendering quality to `LOW`:

  ```
  _quality = "LOW";
  ```

  For more information about `SetQuality` command, see `SetQuality` in *Flash Lite 1.x ActionScript Language Reference*.
- Limit the number of simultaneous tweens. Reduce the number of tweens, or sequence the animation so that one begins when another ends.
- Use transparency (alpha) effects on symbols sparingly because they are CPU-intensive. In particular, it is better to avoid tweening symbols that have alpha levels that are not fully opaque (less than 100%).
- Avoid CPU intensive visual effects, such as large masks, extensive motion, alpha blending, extensive gradients, and complex vectors.
- Experiment with combinations of tweens, key frame animations, and ActionScript-driven movement to produce the most efficient results.
- Test animations frequently on your target devices.

Bitmap graphics

Macromedia recommends optimizing bitmap graphics to 16 bits before importing them into Flash Professional 8. Optimizing your graphics reduces SWF file size and gives you more control over the final output. Also, ensure that bitmaps are imported at the size they need to be in the Flash Lite content. Using larger than required bitmaps results in higher runtime memory requirements.

Bitmap versus vector graphics

When you create content for mobile devices, it is sometimes preferable 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 a mobile phone if it were created as a vector graphic; a bitmap would animate better.

However, using bitmaps produces larger files than using vector images, so you should find the right balance of CPU versus file size and runtime memory requirements in your development.

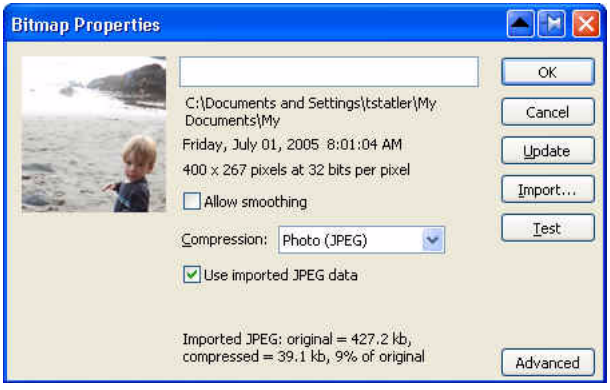
Whenever possible, do not use strokes on your vector shapes, because it increases the number of rendered lines.

Setting bitmap compression options

If you use bitmaps, you can set image-compression options in Flash that educe your SWF file size. You can set bitmap compression on per-image basis, or globally for all bitmap images.

To set bitmap image compression:

1. Start Flash, and create a document.
2. Select a bitmap in the Library window.
3. Right-click (Windows) or Control-click (Macintosh) the bitmap's icon in the Library window, and select Properties from the context menu to open the Bitmap Properties dialog box.



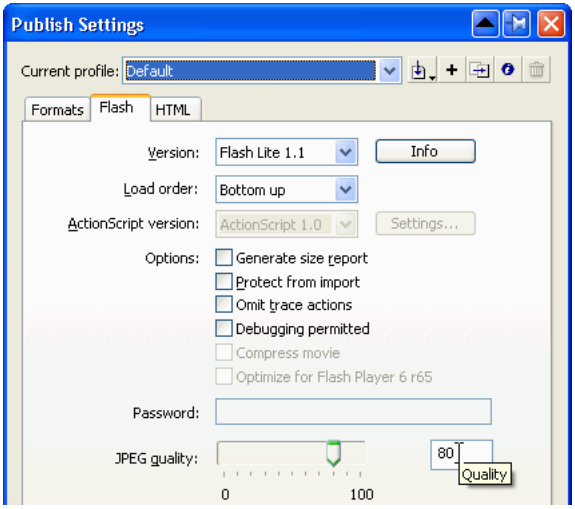
4. In the Compression pop-up menu, select one of the following options:
   - Select the Photo (JPEG) option for images with complex color or tonal variations, such as photographs or images with gradient fills. This option produces a JPEG format file. Select the Use Imported JPEG Data check box to use 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 the Lossless (PNG/GIF) option for images with simple shapes and a few colors. This option compresses the image with lossless compression, in which no data is discarded from the image.
5. Click Test to determine the results of the file compression. Compare the original file size to the compressed file size to decide if the selected compression setting is acceptable.

You can also globally adjust the compression settings for bitmap images.

To globally set bitmap compression for bitmap images:

1. Select File > Publish Settings, and then select the Flash tab.

   The Flash tab displays the compression options.



2. Adjust the JPEG quality slider, or enter a value.

   A higher JPEG quality value provides a higher image quality but a larger SWF file size. A lower image quality produces a smaller SWF file. Try different settings to determine the best trade-off between size and quality.

Optimizing ActionScript

Because of CPU limitations, use the following guidelines when developing ActionScript for Flash Lite content used on mobile 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.
- When possible, avoid string and emulated array processing because it can be CPU-intensive.

Device speed and frames per second

Among devices that support Flash Lite (as of this writing), most play back Flash Lite content at about 8 to12 frames per second (fps). On devices with the slowest processors, the frame rate can be as low as 6 FPS; devices with the fastest processors can achieve a frame rate as fast 15 fps.

During development, Macromedia recommends that you set the document's framerate to best match the playback speed on the target device. This gives you an idea of how your content will run on an actual device with limited performance.

Before publishing your final SWF file, set the document's framerate to at least 20 fps or higher to avoid limiting performance, should the device be capable of a higher framerate.

CHAPTER 5

Testing Flash Lite Content (Flash Professional Only)

5

Macromedia Flash Professional 8 includes a Flash Lite emulator that lets you test your application in the authoring tool as it will appear and function on an actual device. When you're satisfied with the application running in the emulator, you can test it on an actual device. This chapter describes the Flash Lite testing and debugging features that are available in Flash Professional 8.

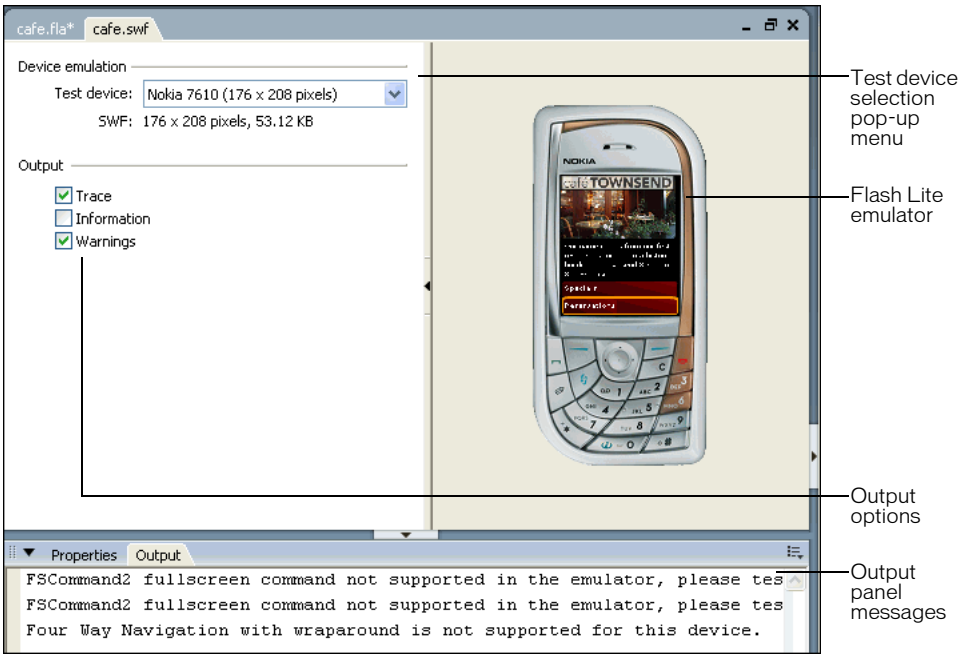
This chapter contains the following topics:

Overview of Flash Lite testing features (Flash Professional Only) 63
Using the Flash Lite emulator (Flash Professional Only). 66
Flash Lite error and warning messages (Flash Professional Only) 74
Selecting test devices and Flash Lite content type (Flash Professional Only). . . . 76
Flash Lite content types (Flash Professional Only) 78
Determining platform capabilities (Flash Professional Only) 81

Overview of Flash Lite testing features (Flash Professional Only)

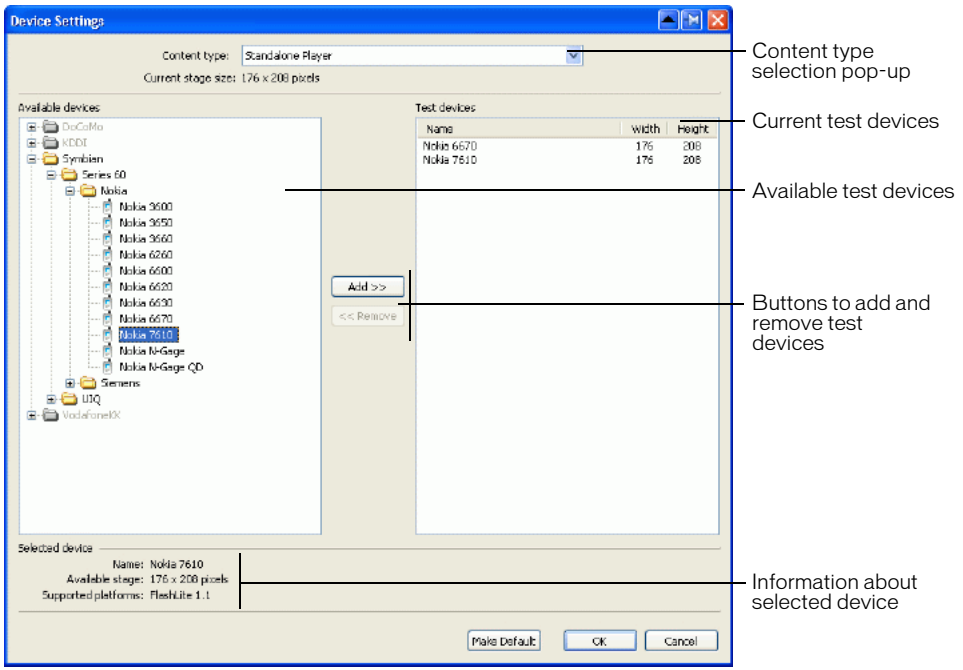
The Flash Lite testing features in Flash Professional 8 consist of two components: the Flash Lite emulator and the Device Settings dialog box. You use the Device Settings dialog box to select which devices are available to you to preview in the Flash Lite emulator.

The Flash Lite emulator lets you preview your Flash Lite content as it will function and appear on an actual device. The emulator also contains controls that let you select a different test device, and change the level of debugging information that the emulator generates. The emulator displays debugging messages in the Output panel to help you troubleshoot problems with your content.



For more information about using the Flash Lite emulator, see "Using the Flash Lite emulator (Flash Professional Only)" on page 66.

The Device Settings dialog box lets you select the test devices and Flash Lite content type that you are targeting. The test devices that you select are available for you to test against in the Flash Lite emulator. Each combination of test device and Flash Lite content type defines a device configuration that specifies what features are available to your application, such as supported audio formats, ability to make network connections, and other features. When you test your application, the emulator displays messages in the Output panel if your application uses features that are not available on the target device.



For more information about the Device Settings dialog box, see "Selecting test devices and Flash Lite content type (Flash Professional Only)" on page 76.

Using the Flash Lite emulator (Flash Professional Only)

The Flash Lite emulator lets you preview your application within the Flash authoring tool.

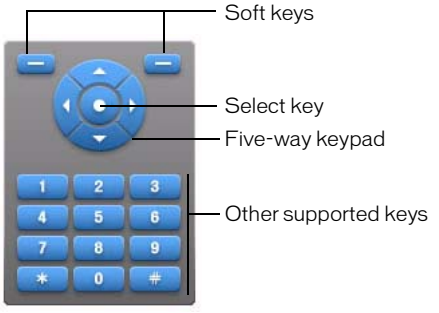
This section contains the following topics:

- "Interacting with the emulator (Flash Professional Only)" on page 66
- "Changing test devices (Flash Professional Only)" on page 67
- "Setting emulator debug options (Flash Professional Only)" on page 69
- "Zooming and rotating the Flash Lite emulator (Flash Professional Only)" on page 70
- "About screen size and available Stage size (Flash Professional Only)" on page 72
- "Testing features not supported in the Flash Lite test window (Flash Professional Only)" on page 72
- "Flash Lite features not supported by the emulator (Flash Professional Only)" on page 73

Interacting with the emulator (Flash Professional Only)

You can interact with the emulator's keypad using your computer mouse, or with keyboard shortcuts. You can interact with the following keys on the emulator's keypad:

- Number keys (0-9), and the asterisk (*) and pound (#) keys.
- Five-way keypad (Left, Right, Down, Up, and Select)
- Left and Right soft keys



You can use your mouse to click directly on the Flash Lite emulator's keypad, or use the following equivalent keyboard shortcuts:

- The arrow keys (Left, Right, Up, Down) map to the arrow keys on the emulator's five-way keypad.
- The Enter or Return key corresponds to the Select key on the emulator's five-way keypad.
- The Page Up and Page Down keys correspond to the Left and Right soft keys, respectively. (Soft keys are only supported by Flash Lite 1.1 and later.)
- The number keys on your keyboard map to the corresponding number keys on the emulator keypad.

For more information about creating interactivity with the keypad, see "Creating Interactivity and Navigation" on page 7.

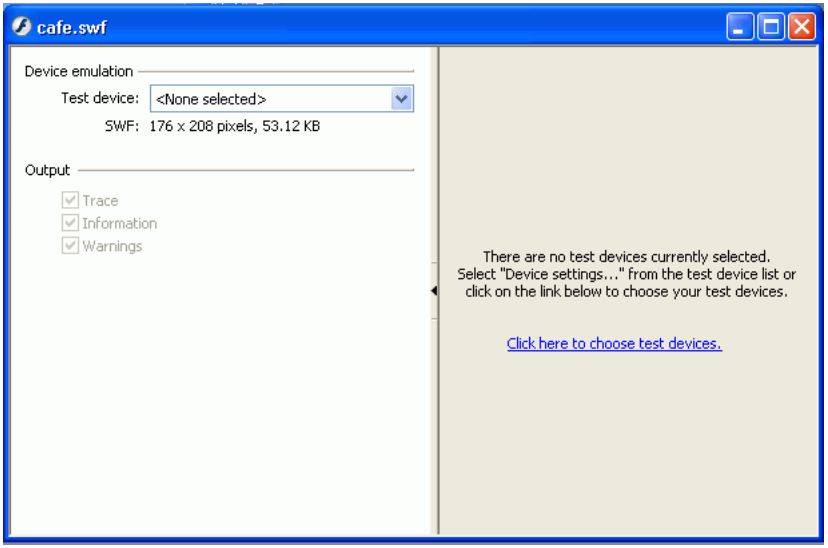
Changing test devices (Flash Professional Only)

The Test Device pop-up menu in the emulator settings pane lets you select a different test device in which to preview your content. The test devices listed in the Test Device pop-up menu are determined by the devices that you selected in the Device Settings dialog box (see "Selecting test devices and Flash Lite content type (Flash Professional Only)" on page 76). When you select a new test device, the Flash Lite emulator reconfigures itself to match the Flash Lite player profile on the target device, and loads a new device skin, if one is available for the selected test device. If a device skin is not available for the specific device, the emulator displays a generic device skin, as the following figure shows:

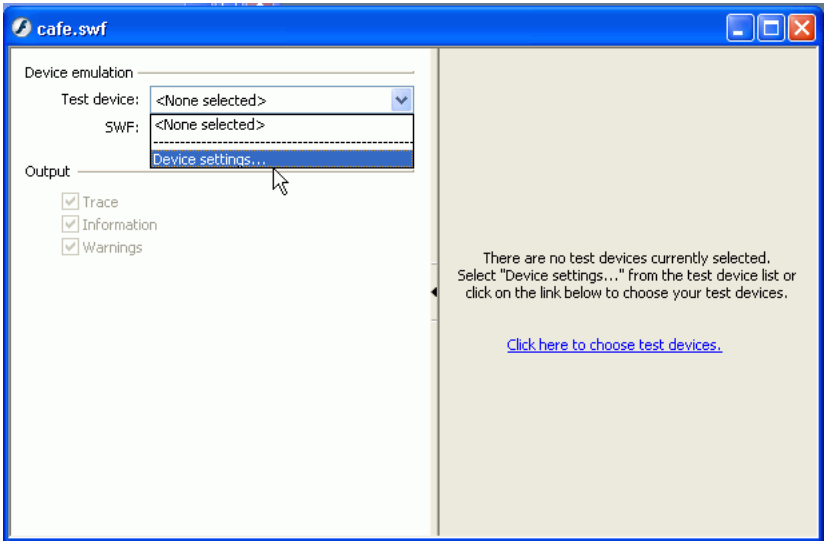
To select a different test device:

1. Open the Flash Lite test window by selecting Control > Test Movie.
2. In the settings pane of the Flash Lite test window, select a new device from the Test Device pop-up menu.

If you haven't yet selected any test devices before testing your application, the Test Device pop-up menu says <None Selected>, and Flash displays a warning message where the emulator normally appears.



To open the Device Settings dialog box and select your test devices without leaving the Flash Lite test window, click the message (in blue text) in the emulator pane, or select the Device Settings option in the Test Device pop-up menu.



For more information about selecting your test devices, see "Selecting test devices and Flash Lite content type (Flash Professional Only)" on page 76.

Setting emulator debug options (Flash Professional Only)

The Flash Lite emulator can send debugging messages to the Output panel while content is running. The emulator reports the following types of information to the Output panel:

Trace messages that are generated by a `trace()` function call within your Flash Lite application. For more information about using `trace()`, see `trace()` in *Flash Lite 1.x ActionScript Language Reference*.

Information messages that contain general information about the selected test device, SWF file size, and other information. For a list of possible information messages, see "Warning and Error Messages (Flash Professional Only)" on page 85.

When the Information debug option is selected, the emulator also outputs the platform capabilities for the selected device and content type. This is useful for determining what features are available to your application on the target platform. For more information, see "Determining platform capabilities (Flash Professional Only)" on page 81.

Warning messages that contain information about problems with your Flash Lite content that might affect playback. For a list of possible warning messages, see "Warning and Error Messages (Flash Professional Only)" on page 85.

The settings pane in the Flash Lite test window contains controls that let you filter the type of information that the emulator generates.

To set Flash Lite output options:

1. Select Control > Test Movie.
2. In the settings pane in the Flash Lite test window, do the following:
   - Select or deselect the Trace option.
   - Select or deselect the Information option.
   - Select or deselect the Warnings option.

Zooming and rotating the Flash Lite emulator (Flash Professional Only)

You can rotate the Flash Lite emulator, as well as zoom in and out on its contents. Rotating the emulator is useful if the content in your application is meant to be viewed in an orientation that is different from the device's normal orientation. For example, suppose you're creating a game for a device whose screen is oriented vertically, but the game is designed to be played with the device oriented horizontally. When you test your application in the emulator, you can rotate the entire emulator to view it as a user would actually view it on the device.

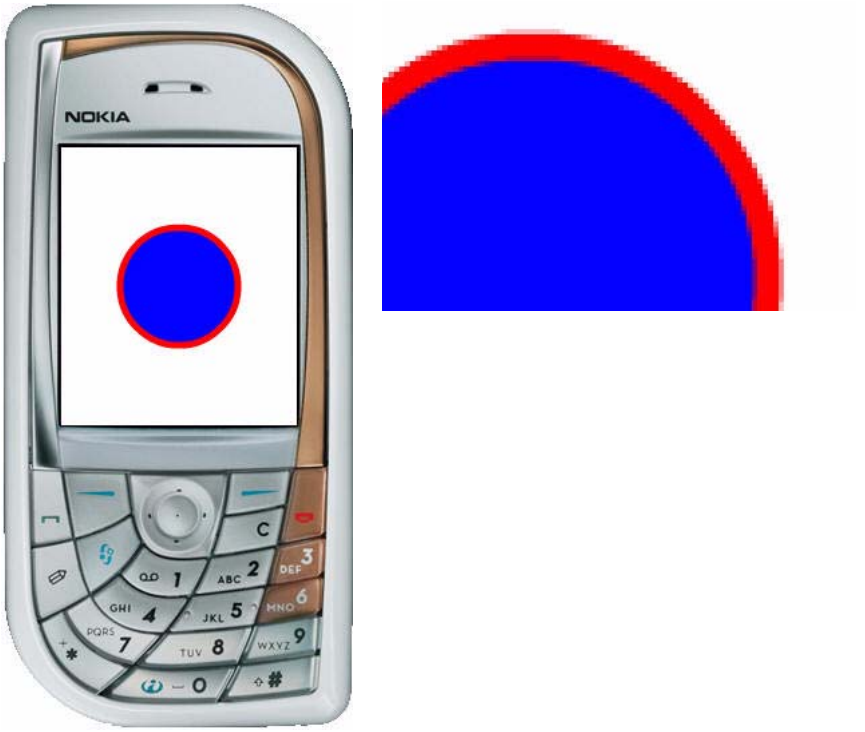
The following image shows the Flash Lite emulator after it has been rotated 90° in the counter-clockwise direction:

To rotate the Flash Lite emulator orientation, do one of the following:

- Select View > Rotate 90° CCW or View > Rotate 90° CW.
- Right-click (Windows) or Command-click (Macintosh) on the emulator and select Rotate 90° CCW or Rotate 90° CW.

When you magnify content in the Flash Lite emulator, it functions as if you held a magnifying glass over the actual device's screen. This causes both vector and bitmap images to appear jagged and pixelated. This behavior is somewhat different than the magnification feature in the standard test Flash Player, which smoothly scales vector graphics as you increase the zoom level.

For example, the following two images show the same content—a simple vector shape. The image on the left show the content as viewed in the Flash Lite emulator at normal magnification level. The image on the right shows a portion of the same content zoomed to 400%.



To zoom in or out on the Flash Lite emulator, do one of the following:

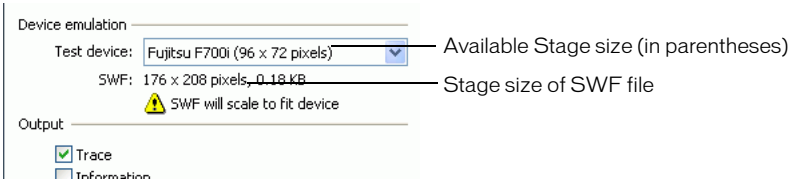
- Select View > Magnification, and then select the desired magnification.
- Select View > Zoom In or View > Zoom Out.
- Press Ctrl+= (to zoom in) or Ctrl+- (to zoom out).

About screen size and available Stage size (Flash Professional Only)

Each combination of target device and Flash Lite content type determines, among other things, the available screen area that a Flash Lite application can occupy. The available Stage area may be equal to, or less than, the device's full screen size.

For example, the Stage area that is available to a SWF file running in full-screen mode in the stand-alone player on a Nokia Series 60 device is equal to the device's full screen size (176 x 208 pixels). On other devices (such as those available in Japan), the Stage area that is available to a SWF file running in one of the specialized content types (such as Address Book or Screensaver) may be less than the device's total screen size. For example, the Fujitsu 700i has a screen size of 240 x 320; however, a SWF file running in the device's Address Book application has 96 x 72 pixels of available Stage area.

If a SWF file's Stage size is different than the available Stage size, the Flash Lite player scales the content (proportionately) to fit within the available Stage area. When you test your content in the Flash Lite emulator, the emulator also warns if your application's Stage size is different from the available Stage area, as the following image shows:



To avoid any undesirable scaling issues, Macromedia recommends that your Flash document's Stage dimensions match the available Stage area for the selected test device and content type.fs

Testing features not supported in the Flash Lite test window (Flash Professional Only)

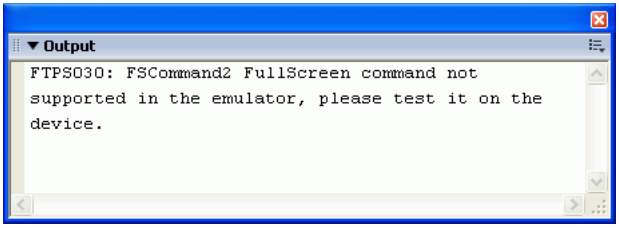
The Flash Lite test window does not support all the features available in the standard (desktop) test window. The following is a list of testing features that are not available when testing your Flash Lite content in the emulator:

- The List Variables (Debug > List Variables) and List Objects (Debug > List Objects) features
- The Bandwidth Profiler, and Streaming and Frame by Frame graphing features
- The View > Simulate Download menu command

- The ActionScript Debugger
- The View > Show Redraw Regions menu command
- The Controller toolbar (Window > Toolbarsf > Controller)

Flash Lite features not supported by the emulator (Flash Professional Only)

The emulator doesn't support all the features that are available to Flash Lite applications running on an actual device. For example, the emulator doesn't support the ability to initiate phone calls or SMS messages. If you attempt to use a command or feature that isn't supported by the emulator, the emulator generates a message in the Output panel, as the following image shows:



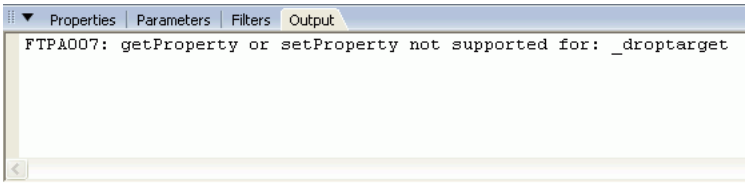
The Flash Lite emulator does not support the following `fscommand()` and `fscommand2()` commands:

- `FullScreen`
- `GetFreePlayerMemory`
- `GetTotalPlayerMemory`
- `Launch`
- `Quit`
- `StartVibrate`
- `GetNetworkConnectStatus`
- `GetNetworkRequestStatus`
- `GetNetworkStatus`

Flash Lite error and warning messages (Flash Professional Only)

There are two types of error messages that the Flash Lite emulator generates while you test your content. One type of message appears only in the emulator; the other occurs in the emulator and on an actual device.

The first type of error message provides useful debugging information about your SWF file. For example, if your SWF contains ActionScript that isn't supported by Flash Lite (or by the version of Flash Lite available on the currently selected test device, the emulator generates a warning message in the Output panel alerting you, as the following image shows:



For a full list of debugging messages and their descriptions, see Appendix, "Flash Lite emulator error and information messages (Flash Professional Only)," on page 85.

The other type of error message that can occur in the emulator also occurs on an actual device. These types of errors are displayed in an error dialog box that the user must close for the application to continue. The following image shows an example error dialog box as it appears in the emulator:

On a device, the error dialog box that appears contains the string "Problem with content" followed by an error number. In the emulator, the error dialog box also contains a short error string. The emulator also displays a longer description of the error in the Output panel.

The following table lists all the errors that occur in the Flash Lite player, including error numbers, the short descriptions that appear in the error dialog box, and the longer descriptions that appear in the Output panel:

Error number	Error string	Description and possible causes
1	Out of Memory.	The emulator has run out of heap memory. Unless otherwise specified, the emulator allocates 1 MB of memory for a SWF file to use.
2	Stack Limit Reached	The emulator has detected that its stack limit has been reached or exceeded. This could be caused by various reasons, including multiple levels of nested movie clips or complicated vector drawings.
3	Corrupted SWF	The emulator has detected that the SWF data is corrupted.
4	ActionScript Stuck.	The emulator has detected that certain ActionScript code in the SWF file is taking too long to execute. As a result the emulator has stopped executing the ActionScript code.
5	ActionScript Processing.	The emulator has detected an ActionScript error, such as a reference to a nonexistent movie clip.
6	ActionScript Infinity Loop.	The emulator has detected an infinite loop or deeply nested ActionScript (for example, deeply nested `if..else` statements).
7	Invalid Frame Buffer	The emulator has detected an invalid frame buffer.
8	Invalid Display Rect	The emulator has detected an invalid display rectangle.
9	Invalid Frame Number	The emulator has detected that the SWF file has attempted to move to or resolve an invalid frame number.
10	Invalid Key	The emulator has detected an invalid key input.
11	Bad JPEG Data	The emulator has detected that JPEG or PNG data in the SWF file is corrupted, there is not enough memory to decode the JPEG data, or the format of the JPEG data is not supported.

Error number	Error string	Description and possible causes
12	Bad Sound Data.	The emulator has detected that the SWF file contains an unsupported sound data format.
13	Root Movie Unloaded.	The emulator has detected that the root movie has been unloaded and was not replaced with another SWF file.

Selecting test devices and Flash Lite content type (Flash Professional Only)

You use the Device Settings dialog box to select the test devices and content type for your Flash Lite application. When you preview your application in the Flash Lite emulator, the emulator mimics the Flash Lite player configuration for the selected device and content type. The Device Settings dialog box is only available when your Flash document is configured to target the Flash Lite 1.0 or Flash Lite 1.1 player.

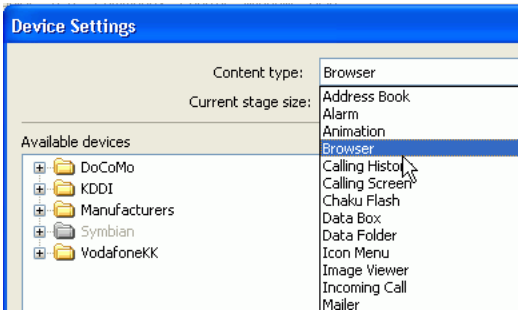
The devices and content types listed in the Device Settings dialog box are loaded from a local XML database that is installed with Macromedia Flash Professional 8. As more devices that support Flash Lite become available, you can download and install an updated list of devices and content types from the Flash Lite Device Updater page at www.macromedia.com/go/mobileupdate. You can also click the Check for New Devices link located at the bottom of the Device Settings dialog box:

To select your test devices and Flash Lite content type:

1. In Flash, open the Device Settings dialog box, by doing one of the following:
   a. Click the Device Settings button in the Property inspector (Window > Properties > Properties).
   b. Select File > Device Settings.

   You can also open the Device Settings dialog box when you test your application in the emulator. For more information, see "Using the Flash Lite emulator (Flash Professional Only)" on page 66.

2. In the Device Settings dialog box, select the content type for your application from the Content Type pop-up menu.

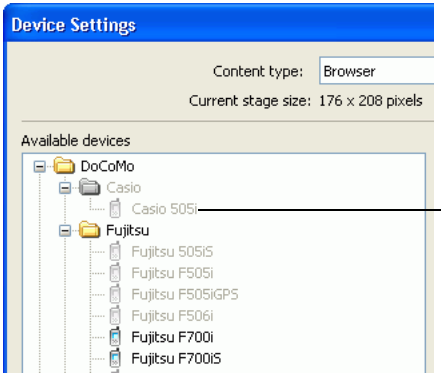


The content type that you select corresponds to the Flash Lite player configuration that your content will run in. For example, if you're creating content for the stand-alone version of Flash Lite, you would select Standalone Player from this menu.

3. Add devices from the list of available devices (on the left) to the list of test devices (on the right).

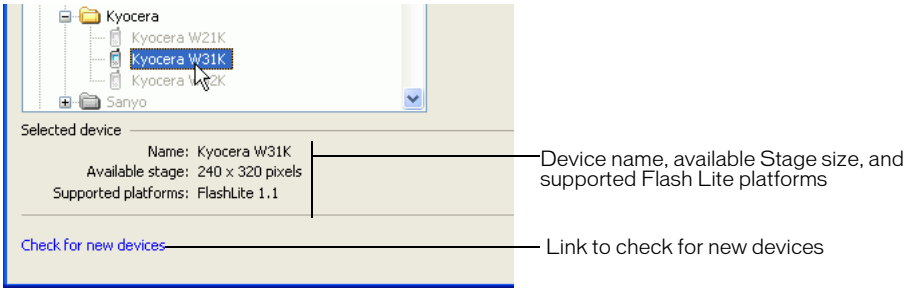
   - To add a single test device, select it and click Add.
   - To select all the devices in a specific folder, select the folder and click Add.

   If a device doesn't support the selected content type, it appears dimmed. (On Macintosh systems, the icon next to the name of each device is dimmed, but not the text itself.) If none of the devices in a folder support the selected content type, the folder itself is dimmed. You can add dimmed devices to your list of test devices, but they won't be available when you test your application in the emulator.



Dimmed devices don't support the selected content type.

4. To view information about a device, select the device in either the Available Devices or Test Devices pane. The lower edge of the Device Settings dialog box displays the device's model name, available Stage area, and supported Flash Lite platforms.



The Available Stage property is the size of the available screen area on that device for the content type. If your SWF file's actual Stage dimensions don't match those of the available Stage area, then Flash Lite scales your content to fit the available screen space. For more information, see "About screen size and available Stage size (Flash Professional Only)" on page 72 in *Getting Started with Flash Lite*.

5. Click Make Default to make the currently selected content type and test devices the default device settings.

   Flash automatically applies the default device settings to any new document whose SWF file's publish version is set to Flash Lite 1.0 or Flash Lite 1.1.

6. Click OK to close the Device Settings dialog box.

Flash Lite content types (Flash Professional Only)

As discussed in "About Flash Lite content types" in *Getting Started with Flash Lite*, Flash Lite is installed on a variety of devices. Each Flash Lite installation supports one or more application modes, or *content types*. For example, some devices use Flash Lite to enable screen savers or animated ring tones that are based on Flash. Other devices use Flash Lite to render Flash content that is embedded in mobile web pages.

The following table lists and describes all the Flash Lite content types available as of this writing. As discussed in "Flash Lite 1.x availability" in *Getting Started with Flash Lite*, most of the content types are only available in a specific geographic region or from specific mobile operators. The third column in the table provides the availability of each content type by region and mobile operator. For additional and up-to-date information about Flash Lite content type availability, see the Flash Enabled Mobile Device page at www.macromedia.com/mobile/supported_devices/.

Flash Lite supports the following content types:

Flash Lite content type	Description	Availability
Address Book	Uses Flash Lite to let users associate a SWF file with an entry in their device's address book application.	NTT DoCoMo and Vodafone (Japan only)
Alarm	Uses Flash Lite to let the user select a SWF file to play for the device's alarm.	KDDI and Vodafone (Japan only)
Browser	Uses Flash Lite to render Flash content embedded in mobile web pages and viewed in the device's web browser.	NTT DoCoMo, KDDI, and Vodafone (Japan only)
Calling History	Uses Flash Lite to display an image or animation associated with each entry in the user's address book, along with their name and phone number.	KDDI (Casio phones only)
Calling Screen	Uses Flash Lite to display an animation when the user receives a call or makes a call.	NTT DoCoMo and KDDI (Japan only)
Chaku Flash	Uses Flash Lite to let user select SWF file to play as the ringtone for incoming calls.	KDDI (Japan only)
Data Box	Uses Flash Lite to render Flash content in the device's Data Box application, which lets the user manage and preview multimedia files on the device.	NTT DoCoMo, KDDI, and Vodafone (Japan only)
Data Folder	Uses Flash Lite to render Flash content in the device's Data Folder application, which lets the user manage and preview multimedia files on the device.	KDDI (Japan only)
Icon Menu	Uses Flash Lite to let the user select custom icon menus for the device's launcher .	KDDI (Casio phones only)

Flash Lite content type	Description	Availability
Image Viewer	Use the Image Viewer application that lets the user manage and preview multimedia files on the device, including SWF files.	NTT DoCoMo (Japan only)
Incoming Call	Uses Flash Lite to display an animation when the user receives a call.	NTT DoCoMo, KDDI, and Vodafone (Japan only)
Mailer	Uses Flash Lite to display an animation when the user sends or receives an e-mail message.	Vodafone (Japan only)
Multimedia	Uses Flash Lite to preview SWF files (as well as other multimedia formats).	KDDI (Japan only)
My Picture	Uses My Picture application that lets the user manage and preview SWF files on the device, as well as other image formats.	NTT DoCoMo (Japan only)
OpenEMIRO	Displays Flash Lite content when the device is returning from Standby mode. This is similar to the Wake Up Screen content type on other devices.	KDDI (Casio devices only)
Screen Saver	Uses Flash Lite to display the device's screen saver.	KDDI and Vodafone (Japan only)
SMIL Player	Uses Flash Lite to preview SWF files (as well as other multimedia formats).	KDDI (Japan only)
Standalone Player	Makes Flash Lite available as a stand-alone application so that the user can launch and view arbitrary SWF files that reside on the device or that the user receives in the messaging in-box.	Available globally for select Symbian Series 60 and UIQ devices.
Standby Screen	Uses Flash Lite to display the device's Standby Screen (or wallpaper screen).	NTT DoCoMo and KDDI (Japan only)
Sub LCD	Uses Flash Lite to display content on the external or secondary screen available on some flip phones.	KDDI (Japan only)
Wake Up Screen	Uses Flash Lite to display an animation as the phone is starting.	NTT DoCoMo (Japan only)

Determining platform capabilities (Flash Professional Only)

Each combination of target device and Flash Lite content type defines a set of available Flash Lite features, such as navigation type, supported device sound formats, or input text support. When the Information debug option is enabled in the emulator settings pane, the emulator generates a list of platform capabilities for the currently selected device and content type. For more information on setting debug options, see "Setting emulator debug options (Flash Professional Only)" on page 69.

The following table describes the Flash Lite platform capabilities as they are reported in the Output panel when you test your application in the emulator:

Capability name	Description and possible values
DeviceSoundKeyOnly	Indicates if the device plays device sounds only in response to the user pressing a key on the device (**Yes**), or independently of any user action (**No**). For more information, see "Triggering device sounds in Flash Lite 1.0" on page 53.
DeviceSoundsOrdered	A comma-delimited list of device sound formats that the platform supports. The order of the sound formats indicates what sound Flash Lite plays if a SWF file contains a sound bundle file with multiple sound formats. For more information about sound bundles, see "Using compound sound (Flash Professional Only)" on page 54.
FSCommand	Indicates how frequently Flash Lite processes `fscommand()` or `fscommand2()` function calls. Valid values are as follows: **OnePerKey**: Only one `fscommand()` call is allowed for each keypress. **OnePerKeyPerFrame**: Only one `fscommand()` call is allowed per event handler or per frame. **All**: No restriction on how frequently `fscommand()` can be called. **None**: The `fscommand()` function is not supported.
InputText	Indicates if platform supports input text (**Yes**) or not (**No**). For more information about input text, see "Using input text fields" on page 34).
LoadMovie	Indicates how frequently Flash Lite processes `loadMovie()` function calls. Valid values are as follows: **OnePerKey**: Only one `loadMovie()` call is allowed for each keypress. **OnePerKeyPerFrame**: Only one `loadMovie()` call is allowed per event handler or per frame. **All**: No restriction on how frequently `loadMovie()` can be called. **None**: The `loadMovie()` function is not supported.

Capability name	Description and possible values
LoadVars	Indicates how frequently Flash Lite processes `loadVariables()` function calls. Valid values are as follows: **OnePerKey**: Only one `loadVariables()` call is allowed for each keypress. **OnePerKeyPerFrame**: Only one `loadVariables()` call is allowed per event handler or per frame. **All**: No restriction on how frequently `loadVariables()` can be called. **None**: The `loadVariables()` function is not supported.
Loop	Indicates if SWF content loops (return to the first frame in the timeline) when it reaches the end of its timeline (**Yes**) or stops on the last frame (**No**).
MultipleDeviceSound	Indicates if the device supports mixing of multiple device sounds (**Yes**) or not (**No**).
NativeSounds	This is a non-ordered list of sound formats the Flash Lite player can play natively (as opposed to device sounds which are passed from the Flash Lite player to the device for playback). Possible values: **NativeSound_PCM**, **NativeSound_ADPCM** and **NativeSound_MP3**.
NavigationType	Indicates the navigation mode supported by the platform: two-way, four-way, or four-way with wrap-around. For more information about navigation modes, see "Modes of tab navigation" on page 9. Valid values are as follows: **2Way**: Up and Down Arrow keys supported only. **4Way**: All four arrow keys (Up, Down, Left, and Right) are supported for navigation. **4WayWrapAround**: Same as 4Way except that focus wraps around to the top of the display.
SMS	Indicates if Flash Lite supports sending SMS messages (**Yes**) or not (**No**).
getUrl	Indicates how frequently Flash Lite processes `getURL()` function calls. Valid values are as follows: **OnePerKey**: Only one `getURL()` call is allowed for each keypress. **OnePerKeyPerFrame**: Only one `getURL()` call is allowed per event handler or per frame. **All**: No restriction on how frequently `getURL()` can be called. **None**: The `getURL()` function is not supported.

Capability name	Description and possible values
keySet	Indicates what key events are supported by Flash Lite on the device. For more information about handling key events, see "Handling key events (Flash Professional Only)" on page 13. Valid values are as follows: **All**: All key events are handled: **Phone**: Only events associated with the 0-9, #, *, Select, and four-way navigation keys are handled.
mouseType	Indicates what mouse events are supported by Flash Lite. Valid values are as follows: **None**: No mouse events are supported. **Partial**: The `press`, `release`, `rollOver`, and `rollout` events are supported; `releaseOutside`, `dragOut`, and `dragOver` events not supported. **Mouse**: The Mouse Up/Mouse Down/Mouse Move messages are processed. One example is an NTT DoCoMo phone with the virtual cursor feature. This means mouse-move should trigger the `rollOver`/`rollOut` event.
soundEnabled	Indicates if sound is enabled on the device (**Yes**) or not (**No**).

APPENDIX

Warning and Error Messages (Flash Professional Only)

This appendix lists the possible information and warning messages that the Flash Lite emulator might generate while your testing your Flash Lite application. For more information about these warning messages, see "Setting emulator debug options (Flash Professional Only)" on page 69.

Also see "Flash Lite error and warning messages (Flash Professional Only)" on page 74 which lists errors that can occur on both the emulator and on an actual device.

Flash Lite emulator error and information messages (Flash Professional Only)

The following table lists all the information messages that the Flash Lite emulator reports:

Error code	Message	Description
FTPA002	FSCommand is ignored.	The emulator detected an `fscommand()` function call, which is not supported by the selected test device. No modifications are made to the device-specific SWF file–this is just a warning.
FTPA003	loadVariables is ignored.	The emulator detected a `loadVariables()` function call, which is not supported by the selected test device and content type. No modifications are made to the device-specific SWF file–this is just a warning.
FTPA004	loadMovie is ignored.	The emulator detected a `loadMovie()` function call, which is not supported by the selected test device and content type. No modifications are made to the device-specific SWF file–this is just a warning.

Error code	Message	Description
FTPA005	The call to GetURL for *URL* was ignored because there was more than one request per keypress.	Flash Lite allows only one `getURL()` function call per keypress; the emulator detected that there was more than one `getURL()` 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.	The currently selected test device and content type only processes `getURL()` function calls that result from the user pressing a key on their device. The emulator detected that your application made a call to `getURL()` that wasn't associated with a keypress.
FTPA007	getProperty or setProperty not supported for: *movieclip property*.	Flash Lite does not support the specified movie clip property.
FTPA008	getProperty or setProperty not fully supported for: *movieclip property*.	Flash Lite does not fully support the specified movie clip property. For more information, see the entry for the specified property in *Flash Lite 1.x ActionScript Language Reference*.
FTPA009	startDrag and stopDrag are not supported.	The emulator detected a `startDrag()` or `stopDrag()` function call, which Flash Lite does not support.
FTPA014	getURL is ignored.	The emulator detected a `getURL()` function call, which is not supported by the selected test device and content type. No modifications are made to the device-specific SWF file–this is just a warning.
FTPA015	The call to loadMovie for *URL* was ignored because there was more than one request per keypress.	Flash Lite allows only one `loadMovie()` function call per keypress; the emulator detected that there was more than one `loadMovie()` so only the first command is processed–the others are ignored.
FTPA016	The call to loadMovie for *URL* was ignored because it was not associated with a keypress.	The currently selected test device and content type only processes `loadMovie()` function calls that result from the user pressing a key on their device. The emulator detected that your application made a call to `loadMovie()` that wasn't associated with a keypress.

Error code	Message	Description
FTPA017	The call to loadVariables for *URL* was ignored because there was more than one request per keypress.	Your application made multiple `loadVariables()` function calls during a single keypress event. Flash Lite allows only one `loadVariables()` command per keypress, so only the first command is processed–the others are ignored.
FTPA018	The call to loadVariables for *URL* was ignored because it was not associated with a keypress.	The currently selected test device and content type only processes `loadVariables()` function calls that result from the user pressing a key on their device. The emulator detected that your application made a call to `loadVariables()` that wasn't associated with a keypress.
FTPA019	The call to FSCommand with arguments *command-arguments* was ignored because there was more than one request per keypress.	Flash Lite allows only one `fscommand()` function call per keypress; the emulator detected that there was more than one `fscommand()` so only the first command is processed–the others are ignored.
FTPA020	The call to FSCommand with arguments *command-arguments* was ignored because it was not associated with a keypress.	The currently selected test device and content type only processes `fscommand()` function calls that result from the user pressing a key on their device. The emulator detected that your application made a call to `fscommand()` that wasn't associated with a keypress.
FTPE001	The key will not be processed: *keyname* ASCII Value: *value*	The emulator detected that a device key was pressed that isn't supported by Flash Lite–the keypress is ignored.
FTPE013	Input text fields are not supported for the selected content type on this device.	The current test device and content type does not support input text fields.
FTPS010	Streaming Sound is unsupported.	The selected test device and content type do not support streaming sound.
FTPS011	Only a single sound can be played at a time (no mixing).	The emulator detected that the SWF file contains multiple sounds playing simultaneously, which is not supported in Flash Lite.

Error code	Message	Description
FTPS012	Event sound was ignored because it was not associated with a keypress.	In Flash Lite 1.0, a sound can play only in response to the user pressing a key on their device. For more information, see "Triggering device sounds in Flash Lite 1.0" on page 53.
FTPS021	Sound not supported for the selected content type on this device.	The selected test device and content type do not support sound.
FTPS022	ADPCM sounds not supported for the selected content type on this device.	The emulator detected that the SWF file contains a native (nondevice) sound compressed with ADPCM compression, which is not supported by the selected content type on this device. No modifications are made to the device-specific SWF file–this is just a warning.
FTPS023	MP3 sounds not supported for the selected content type on this device.	The emulator detected that the SWF file contains a native (nondevice) sound compressed with MP3 compression, which is not supported by the selected content type on this device. No modifications are made to the device-specific SWF file–this is just a warning.
FTPS024	MIDI sounds not supported for the selected content type on this device.	The emulator detected a MIDI device sound, which is not supported by the selected content type on this device.
FTPS025	PCM sounds not supported for the selected content type on this device.	The emulator detected a native Flash sound compressed using PCM compression, which is not supported by the selected content type on this device.
FTPS026	Debug movie is not supported in the specified test movie player.	The Flash Lite emulator does not support the Control › Debug Movie menu command.
FTPS027	Sound Bundle found.	The emulator detected that the SWF file contains a sound bundle file.
FTPS028	Invalid FSCommand2 *command-name* command found.	The specified `fscommand2()` command is not a valid command string. For a list of valid `fscommand2()` commands, see Chapter 5, "Flash Lite Specific Language Elements" in the *Flash Lite 1.x ActionScript Language Reference*.

Error code	Message	Description
FTPS029	FSCommand2 *command-name* command found.	The emulator detected the specified `fscommand2()` command.
FTPS030	FSCommand2 *command-name* command not supported in the emulator, please test it on the device.	The emulator does not support the specified `fscommand2()` command. You need to test this SWF file on a device with Flash Lite installed to see if the specified command functions as expected,
FTPS031	More than one instance of URL Request calls found, only one allowed per keypress/ frame	Flash Lite allows only one `getURL()` function call per keypress or frame; the emulator detected that there was more than one `getURL()` so only the first command is processed–the others are ignored.
FTPS032	A call to GetURL(*URL*) found, limitations might apply.	The emulator detected a `getURL()` function call, which may have some runtime restrictions when played on the selected device. Test your SWF file on an actual device to see if the command functions as expected.
FTPS033	A call to loadVariables(*URL*) found, limitations might apply.	The emulator detected a `loadVariables()` function call, which may have some runtime restrictions when played on the selected device. Test your SWF file on a device to ensure that the command functions as expected.
FTPS034	A Call to FSCommand(*command-name*) found, limitations might apply.	This is just a warning that not all devices and Flash Lite content types may support the `fscommand()` in the application. Test your SWF file on a device to ensure that the command functions as expected.
FTPS035	A call to loadMovie(*URL*) found, limitations might apply.	The emulator detected a `loadMovie()` function call, which may have some runtime restrictions when played on the selected device. Test your SWF file on a device to ensure that the command functions as expected.
FTPS036	*N* kilobytes of *device-sound* sound found in sound bundle.	For each sound in a sound bundle, the emulator reports the type (for example, MIDI or SMAF) and size of each sound in the bundle.

Error code	Message	Description
FTPS037	SMAF sounds not supported for the selected content type on this device.	The emulator detected a SMAF device sound, which is not supported by the selected content type on this device.
FTPS038	The call to `StartVibrate` was ignored because there was more than one request per frame or event.	Flash Lite allows only one `fscommand2("StartVibrate")` call per keypress or frame; the emulator detected more than one, so only the first command is processed–the others are ignored.
FTPS039	FSCommand2 SetInputTextType(*command-arguments*) found, not supported in the emulator, please test it on the device.	The `SetInputTextType` command is not supported in the emulator. You must test it on an actual device.
FTPS048	Four Way Navigation is not supported for this device.	The currently selected test device and content type supports two-way navigation. You pressed the Left or Right Arrow keys on the emulator's five-way keypad, which aren't supported in two-way navigation. For more information, see "Modes of tab navigation" on page 9.
FTPS049	Four Way Navigation with wraparound is not supported for this device.	The currently selected test device and content type supports four-way navigation. You pressed one of the device's arrow keys when there were no objects on the Stage to receive focus in the direction of the arrow key that you pressed. For more information, see "Modes of tab navigation" on page 9.
FTPS050	Generic MFI sounds not supported for the selected content type on this device.	The emulator detected a Generic MFI device sound, which is not supported by the selected content type on this device.
FTPS051	Unsupported Mouse Event (*event-name*) found.	The specified mouse event is not supported by the selected test device and content type.
FTPS067	SMAF(MA-2) sounds not supported for the selected content type on this device.	The emulator detected a SMAF (MA-2) device sound, which is not supported by the selected content type on this device.
FTPS068	SMAF(MA-3) sounds not supported for the selected content type on this device.	The emulator detected a SMAF (MA-3) device sound, which is not supported by the selected content type on this device.

Error code	Message	Description
FTPS069	SMAF(MA-5) sounds not supported for the selected content type on this device.	The emulator detected a SMAF (MA-5) device sound, which is not supported by the selected content type on this device.
FTPS070	MFI sounds with Fujitsu extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with a Fujitsu extension, which is not supported by the selected content type on this device.
FTPS071	MFI sounds with Mitsubishi extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with a Mitsubishi extension, which is not supported by the selected content type on this device.
FTPS072	MFI sounds with NEC extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with an NEC extension, which is not supported by the selected content type on this device.
FTPS073	MFI sounds with Panasonic extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with a Panasonic extension, which is not supported by the selected content type on this device.
FTPS074	MFI sounds with Sharp extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with a Sharp extension, which is not supported by the selected content type on this device.
FTPS075	MFI sounds with Sony extension not supported for the selected content type on this device.	The emulator detected a MFI device sound with a Sony extension, which is not supported by the selected content type on this device.
FTPS099	Print Commands are not supported.	Your application contains one of the ActionScript print commands (for example, `print()` or `printAsBitmap()`), which are not supported by Flash Lite.
FTPS100	*Device-sound* sound is chosen in sound bundle.	Indicates the name of the device sound in a sound bundle that the emulator chose to play.
FTPS101	None of the formats in the sound bundle are supported on this device.	Indicates that none of the device sounds in a sound bundle are supported by the selected content type on this device.
FTPS102	SMAF sound playback failed.	The emulator was not able to play the SMAF sound.

Error code	Message	Description
FTPS105	This SWF is not in Flash Lite format.	Your application attempted to load a SWF file whose version was not in the Flash Lite format; Flash Lite can load other Flash Lite SWF files or Flash 4-formatted SWF files, only.
FTPS106	Mouse Event (*event-name*) was ignored because it was not triggered by Keypress.	The emulator detected a mouse event over a button in your Flash Lite application. The current test device does not support a stylus or touch-screen interface, so you can only interact with buttons on the screen using the emulator's keypad or equivalent keyboard shortcuts.
FTPS107	The key will not be processed: *device-key*. Use FSCommand2 SetSoftKeys to enable this key	You pressed one of the soft keys on the emulator's keypad without first calling the `SetSoftKeys` command. For more information, see "Using the soft keys (Flash Professional Only)" on page 26.
FTPS108	Invalid FSCommand (*command-name*) found.	The specified `FSCommand()` command is not a valid command string.
FTPS109	FSCommand (*command-name*) not supported in the emulator, please test it on the device.	The emulator does not support the specified `FSCommand()` command. You must test this SWF file on a device with Flash Lite installed to see if the specified command functions as expected.
FTPS110	Soft keys are not supported in the FlashLite 1.0 player.	The emulator detected that you pressed one of its soft keys but your document's SWF file's version publish setting is set to Flash Lite 1.0. Flash Lite 1.0 does not support soft keys.

Index

A

application modes, in Flash Lite 78

B

button events
  about 20
  handling 20

C

content types in Flash Lite, described 78

D

device settings
  about 76
  selecting test devices and content type 76
device sound
  about 50
  adding 51
  triggering in Flash Lite 1.0 53

E

embed font outlines
  about 40
  how to 40
emulator. *See* Flash Lite emulator

F

Flash Lite emulator
  about 63
  changing test devices 67
  debug options 69
  features unsupported by 72
  interacting with 66
  previewing applications with 66
  rotating 70
  warning and error messages 74, 85
  zooming 70
Flash Lite rendering quality
  and text fields 39
  default rendering quality 39
font rendering methods
  about 38
  applying to text fields 38

I

input text fields
  and the focus rectangle 37
  example application 42
  restricting characters in 37
interactivity
  creating a menu with buttons 22
  creating a menu with movie clips 16
  creating, with buttons 20
  detecting keypresses 14
  handling keypress events 13
  tab navigation 9
  using the soft keys 26

K

keypress events
  ActionScript key codes 14
  creating a key catcher button 14
  handling with ActionScript 13
  supported keys 8
  writing a key handler script 14

M

menus
creating with buttons 22
creating with movie clips 16

N

native sound, about 56
navigation. *See* tab navigation

O

optimizing
ActionScript 61
animation performance 58
bitmap graphics 58, 59
bitmap versus vector graphics 59
file size 57
frames per second 61
memory 57
performance 57

P

platform capabilities, determining 81

S

soft keys
SetSoftKeys command 26
using 26
sound
about 49
compound 54
device 50
device and native 49
event and streaming 50
Sound Bundler utility 54
Sound Bundler utility, using for compound sounds 54
Stage
available 72
screen size and 72

T

tab navigation
about 9
example application using 22
focus rectangle 11
four-way 10
four-way with wrap-around 10
guidelines for 12
modes of 9
two-way 10
text fields
creating scrolling text 45
font rendering methods, about 32
input text fields, using 34
restricting characters in input text fields 37