DEVELOPING FLASH® LITE™ 2.x APPLICTIONS

Adobe®

Developing Flash® Lite™ 2.x Applications

Third-Party Information

This guide contains links to third-party websites that are not under the control of Adobe Systems Incorporated, and Adobe Systems Incorporated is not responsible for the content on any linked site. If you access a third-party website mentioned in this guide, then you do so at your own risk. Adobe Systems Incorporated provides these links only as a convenience, and the inclusion of the link does not imply that Adobe Systems Incorporated endorses or accepts any responsibility for the content on those third-party sites.

Sorenson™ Spark™ video compression and decompression technology licensed from Sorenson Media, Inc.

Fraunhofer-IIS/Thomson Multimedia: MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and Thomson Multimedia (http://www.iis.fhg.de/amm/).

Independent JPEG Group: This software is based in part on the work of the Independent JPEG Group.

Nellymoser, Inc.: Speech compression and decompression technology licensed by Nellymoser, Inc. (http:www.nelly-moser.com).

Opera ® browser Copyright © 1995-2002 Opera Software ASA and its suppliers. All rights reserved.

Macromedia Flash 8 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. http://www.on2.com.

Visual SourceSafe is a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

Updated Information/Additional Third Party Code Information available at http://www.adobe.com/go/thirdparty/.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

CHAPTER 1

Flash Lite 2.x Overview

1

Adobe® Flash® Lite™ is a version of Flash Player designed for devices. This documentation covers Macromedia® Flash® Lite™ 2.0 and Macromedia® Flash® Lite™ 2.1 software from Adobe, which are collectively called Flash Lite 2.x.

Flash Lite 2.x is based on Macromedia® Flash® Player 7 and supports most—but not all—features in Flash Player 7. Flash Lite 2.x also includes features specific to mobile development that are not available in Flash Player 7. For example, in Flash Lite 2.x, you can load device-specific media types (images, sounds, video) that aren't natively supported by Flash Lite.

Flash Lite 2.x also includes device integration features, such as the ability to make phone calls and send text messages.

This chapter provides an overview of the new features in Flash Lite 2.x and discusses the features in Flash Player 7 that Flash Lite doesn't support. This chapter contains the following topics:

About Flash Lite 2.x features

Macromedia® Flash® Lite™ 1.0 and Macromedia® Flash® Lite™ 1.1 are based on Macromedia® Flash® Player 4. Flash Lite 2.0 and 2.1 (collectively called 2.x) are based on Flash Player 7 and support most of the features available in that version of Flash Player, including XML processing and ActionScript 2.0. Flash Lite 2.x also provides several features, not available in Flash Player 7, that are designed for mobile applications. The rest of this section describes the new features in Flash Lite 2.1 and 2.0.

Flash Lite 2.0 new features

Flash Lite 2.0 includes the following new features :

- ActionScript 2.0
- Device video playback
- Local, persistent data storage (Flash Lite shared objects)
- Support for loading device-specific sound and image formats
- New system capabilities information
- Support for additional device keys, including QWERTY keyboard support and support for up to 11 soft keys
- Rich text formatting (partial support)
- Ability to control backlight duration and set custom focus rectangle color
- Support for ActionScript 2.0, which lets you use advanced programming techniques including classes, interfaces, and strict data typing
- Synchronized device sound
- XML processing support

The following features in Flash Player 7 are not available in Flash Lite 2.0:

- Several ActionScript classes available in Flash Player 7 are unsupported or partially supported in Flash Lite 2.0. For more information about available ActionScript, see *Introduction to Flash Lite 2.0 ActionScript.*
- Socket communication using the XMLSocket class (available in Flash Lite 2.1)
- Support for communication with Flash Media Server
- Remote shared objects (local shared objects are partially supported)
- Native support for Flash Video (FLV) playback
- Support for Flash Application Protocol (the binary data communication protocol used by Flash Remoting)
- Cascading Style Sheet (CSS) formatting with text fields
- Masking with device fonts
- Bitmap smoothing while rendering at high-quality

Flash Lite 2.1 new features

Flash Lite 2.1 adds support for inline text input. In previous versions of Flash Lite, input text fields used the device's generic input text dialog box. In Flash Lite 2.1, input text fields can be edited directly.

NOTE	Devices that support complex languages (for example, Hebrew, Arabic, Farsi, and some Asian languages) do not support inline text input. The functionality of input text fields on these devices will be the same for Flash Lite 2.1 as it is for Flash Lite 2.0.

Flash Lite 2.1 enables socket communication using the XMLSocket class on devices that support it. (For more information, see the *Flash Lite 2.x ActionScript Language Reference*.) For more information about input text fields, see "Working with Text and Fonts" on page 37.

Flash Lite 2.1 also adds tools to develop Flash Lite applications that run on devices using the Binary Runtime Environment for Wireless® (BREW®) platform made by QUALCOMM Incorporated. For more information about these tools, see Chapter 5, "Developing Flash Lite Applications for BREW," on page 81.

Flash Lite 2.0 ActionScript

Flash Lite 2.0 ActionScript is the scripting language used in Flash Lite 2.0 and Flash Lite 2.1 applications. It shares some, but not all, of the ActionScript used in Flash Player 7. Flash Lite 2.0 also includes several ActionScript additions and extensions that let you, for example, get information about the device, make phone calls, or control the backlight duration.

You can use ActionScript 2.0 or ActionScript 1.0 syntax when you develop applications for Flash Lite 2.0. ActionScript 2.0 provides authoring support for classes, interfaces, and strict data typing. Using ActionScript 2.0 syntax lets the ActionScript compiler provide better debugging information, and also encourages better program design.

For more information about learning Flash Lite 2.0 ActionScript, see the following books and topics:

- *Introduction to Flash Lite 2.x ActionScript*
- *Flash Lite 2.x ActionScript Language Reference*
- The "Learning ActionScript 2.0" topic in *Using Flash*

Device video playback

Flash Lite 2.0 can play video in any format that's supported natively by the target device. For example, some devices record and playback video in the 3GP video format. Others support AVI or MPEG video. During playback, Flash Lite passes the original video data to the device to decode and render the data directly to the screen. You can incorporate video in your application in any of the following ways:

- Bundle the original video data in the SWF file.
- Load an external video file from the device's memory card, or over the network.

To control video playback in Flash Lite 2.0 you use the ActionScript Video object. First available in Flash Player 6, the Video object in Flash Lite 2.0 has additional methods for controlling video, as the `Video.play()` and `Video.pause()` methods. You can also use the `System.capabilities.videoMIMETypes` array to determine what video formats a device supports.

For more information about the Video object and using video in Flash Lite, see "Using device video" on page 68.

Loading device-specific sound and image formats

In Flash Lite 2.0, you can load any image or sound file that's in a format supported by the device. To load external images, you use the `loadMovie()` global function, or the `MovieClip.loadMovie()` method. For example, if the device supports the PNG file format, then you could use the following code to load a PNG file from a web server into the movie clip instance image_mc:

```
image_mc.loadMovie("http://www.adobe.com/images/mobile.png");
```

To load external sounds, you use the `Sound.loadSound()` method. In Flash Lite 2.0, you can use this method to load any sound format that the device supports (for example, MIDI or SMAF). External device sounds must fully load into memory before they can play.

As in Flash Lite 1.x, in Flash Lite 2.0 you can also play device sound that's bundled in the SWF file. For more information, see "Using bundled device sound" on page 58.

For more information about loading external images and sounds, see the following topics:

- "Loading external images" on page 79
- "Playing external device sounds" on page 63

Flash Lite shared objects

Flash Lite shared objects let you save data persistently to the user's device. For example, you might use a shared object to save information between application sessions, such as user preferences or game scores. You use the SharedObject class to read and write Flash Lite shared objects. For more information about using Flash Lite shared objects, see the `SharedObject` class in the *Flash Lite 2.x ActionScript Language Reference*.

NOTE	The Flash Lite 2.0 implementation of shared objects does not allow multiple SWF files to share the same data. Also, Flash Lite 2.0 does not support remote shared objects with Flash Media Server.

Synchronized device sound

In previous versions of Flash Lite you could only synchronize native Flash sound to animation in the timeline. But this was not possible with device sounds, which are played directly by the device, rather than natively by Flash Lite. In Flash Lite 2.0, you can synchronize device sound with the timeline using the new `_forceframerate` property. When this property is set to `true`, Flash Lite drops frames as necessary from animation to maintain the frame rate specified in the SWF file. For more information, see "About synchronizing device sounds with animation" on page 64 and the `_forceframerate property` in the *Flash Lite 2.x ActionScript Language Reference*.

Text features

The following features related to text handling are new in Flash Lite 2.0:

- All text in Flash Lite 2.0 is Unicode-based.
- Flash Lite 2.0 provides partial support for HTML formatting and the TextFormat ActionScript class.

For more information about working with text fields in Flash Lite 2.x, see "Working with Text and Fonts" on page 37.

Additional key support

Flash Lite 2.0 provides additional support for device keys, including support for QWERTY keyboards, and up to 12 soft keys (including the standard left and right soft keys).

About components in Flash Lite

The components that install with Flash (for example, DataGrid and Accordion) were designed for use in Flash desktop applications. The memory requirements and processing power that they require typically prohibit their use in Flash Lite applications. Adobe recommends that you don't use the standard user interface components in your Flash Lite applications.

CHAPTER 2

Creating Interactivity and Navigation

2

To interact with your Adobe Flash Lite application, a user must be able to determine which object on the screen currently has focus, navigate among objects, and initiate an action by selecting an object or another key. While these basic principles are the same as for desktop applications, some of the functionality varies for mobile devices.

This chapter contains the following topics:

About user interaction in Flash Lite

Flash Lite supports navigation and user interaction through the device's keypad, or through a stylus or touch-screen interface on devices that provide one. The options available to your application vary depending on the target device and content type. For more information about content types, see "About Flash Lite content types" in *Getting Started with Flash Lite 2.x*.

The simplest way to add key-based interactivity to a Flash Lite application is default navigation, which uses the device's four-way keypad like the arrow keys or the Tab and Shift+Tab keys in a desktop application. The user moves the focus to the desired object and then presses the select key. The application includes event handler code to respond to these button events. Default navigation in Flash Lite works with buttons, input text fields, and, optionally, movie clips; it is typically best for simple user interactions such as menus. For more information about default navigation, see "Using default navigation in Flash Lite" on page 15.

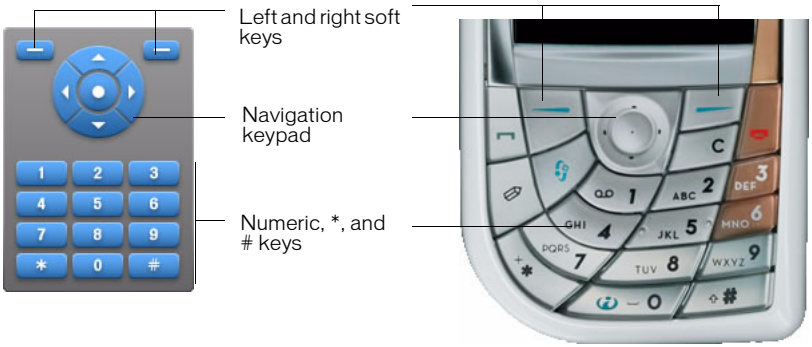
Applications can also respond to arbitrary key press events that Flash Lite generates when a user presses a particular key. Event-based navigation allows you to create Flash Lite applications like games that have a complex user interaction model. For more information about events, see "Handling key and button events" on page 23.

Keys supported by Flash Lite

In addition to the alphanumeric keys available on standard telephones, most mobile devices feature a navigation keypad, which let users navigate and select items on the device screen, and two (or more) soft keys. A device's *soft keys* are multifunctional keys that use the screen to identify their purpose at any moment.

A typical navigation keypad has four navigation keys (up, down, left, and right) and a select key (typically located at the center of the keypad). Different applications can use these keys in different ways. For example, a game might use the navigation keys to let the user move a character on the screen, and then use the select key to perform another action, such as make the character jump.

The following images show the most common keys on a generic keypad and on an actual device:



Not all devices and Flash Lite content types support all these keys. For example, devices that support two-way navigation don't support the left and right navigation keys (see "Default navigation modes" on page 16). Also, not all devices have access to the device's soft keys.

Flash Lite supports the following keys on mobile devices:

Description	Keys	Availability
Numeric, *, #	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, *, #	All devices
Five-way keypad	Select, up, and down	All devices
	Left and right	Devices that support four-way navigation only (see "Default navigation modes" on page 16)
Soft keys	Left and right	Devices that support the `SetSoftKeys` command

Description	Keys	Availability
	SOFT3 - SOFT12 keys	Devices that have more than two soft keys
Keyboard keys	!, “, #, $, %, &, ‘, (,), *, +, ,, -, ., /, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, :, ;, <, +, >, ?, @, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, [, \,], ^, _, ‘, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, {, |, }, ~, Backspace	Devices that have a QWERTY keyboard

The following System.Capabilities properties allow you to determine the navigation and selection options available on a device:

- `hasMappableSoftKeys`
- `softKeyCount`
- `has4WayKeyAS`
- `hasQWERTYKeyboard`
- `hasStylus`
- `hasMouse`

For more information about the System.Capabilites class, see *Flash Lite 2.x ActionScript Language Reference*.

Using default navigation in Flash Lite

On desktop Flash applications, the Tab and Shift+Tab keys let users switch focus among objects on the screen. The arrow keys function in a similar way in some other applications. In Flash Lite, the navigation keys on the device's navigation keypad serve the same purpose as the arrow or Tab and Shift+Tab keys in a Flash desktop application. After the desired object has focus, the user can press the select key to trigger an action in the application. You define event handlers to respond when a button or movie clip is selected; for more information, see “Handling button events” on page 23.

Default navigation in Flash Lite works with buttons and input text fields. Movie clips are also included if their `tabEnabled` property is set to `true`, or if they have event handlers associated with them and their `tabEnabled` property is not set to `false`.

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.

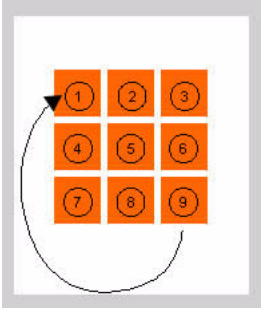
For an example of using default navigation, see “Creating a simple menu using buttons and default navigation” on page 26.

Default navigation modes

Flash Lite supports three modes of default navigation: two-way, four-way, and four-way with wraparound. 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 "About playing a device video in the emulator" on page 126.

Two-way navigation in Flash Lite is analogous to tab navigation in Flash desktop applications, where the Tab and Shift+Tab keys navigate among objects on the screen. The down navigation key on the device corresponds to the Tab key, and the up navigation key corresponds to the Shift+Tab key.

The default tab order in two-way navigation is generally left-to-right and top-to-bottom. For example, the following image shows a three-by-three grid of Button objects in a Flash Lite application. The numbers above each button indicate the order in which each button's will get keypad focus as the user presses their device's down navigation key repeatedly. After the button in the bottom-right corner has received keypad focus, the focus "wraps around" to the top-left button the next time the user presses the down navigation key.



Example tab order in two-way navigation

You can customize the tab order in two-way navigation using the `tabIndex` property of the Button, MovieClip, and TextField objects. For more information, see "About controlling tab order in two-way navigation" on page 22.

For an example of two-way navigation, see the Flash Lite Samples and Tutorials page at www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample file named 2-way.fla.

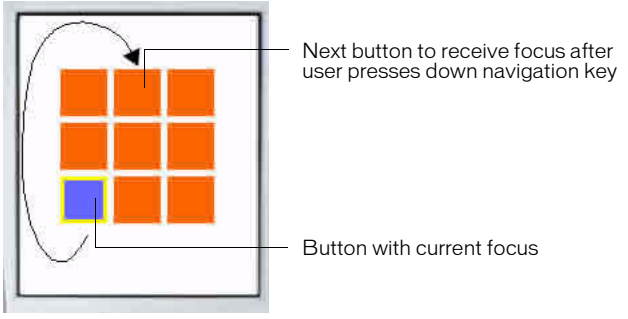
Four-way navigation in Flash Lite is similar to using the arrow keys on a desktop computer's keyboard to navigate among objects on the screen. The device's up, down, left, and right navigation keys correspond to the four arrow keys on a computer's keyboard. Pressing a navigation key moves the keypad focus to the object located in that direction, if one exists. If no object exists in that direction, then the keypad focus does not change from the current object.

NOTE

The `tabIndex` property is not supported on devices that support four-way navigation, but `tabEnabled` and `tabChildren` are, which is different from how these properties work in Flash desktop applications.

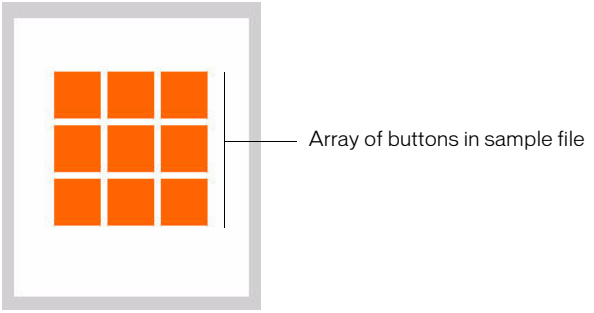
For an example that uses four-way navigation, see the Flash Lite Samples and Tutorials page at www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample file named 4-way.fla,

Four-way navigation with wraparound functions like a combination of standard four-way navigation and two-way navigation. Like standard four-way navigation described previously, users change keypad focus using the device's four-way navigation keys. The difference is that, similar to two-way navigation, keypad focus "wraps around" to the object on the opposite side of the screen. For example, in the image below, the button with the current keypad focus is located at the bottom-left corner of the screen. If the user presses the down navigation key, the next button to receive focus is located in the middle of the top row of buttons.



You can test the behavior of two-way and four-way navigation modes in the Adobe Device Central emulator using the samples named 2-way.fla and 4-way.fla located at www.adobe.com/go/learn_flt_samples_and_tutorials. On the Samples and Tutorials page, locate, download and decompress the .zip file for your Flash Lite version, and then navigate to the Samples folder to access the samples. Each sample file consists of the same three-by-three grid of buttons, as discussed previously. The only difference between the sample files is that each FLA file is configured to target a combination of device and Flash Lite content type that supports the navigation mode (two-way or four-way).

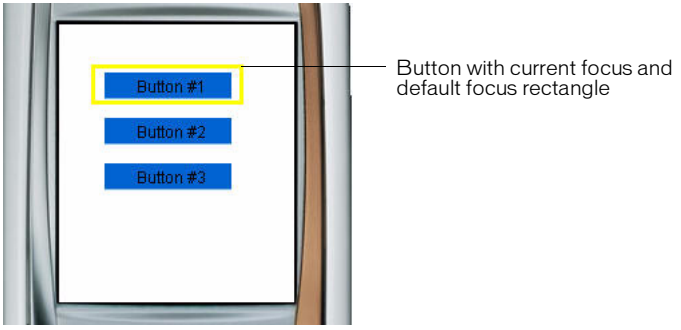
To use each sample file, open it in Flash and test it in the Adobe Device Central emulator (select Control > Test Movie). Click the arrow keys on the emulator's keypad (or press the arrow keys on your keyboard) to see how each navigation mode affects user navigation.



About the focus rectangle

By default, Flash Lite draws a yellow rectangle around the button or input text field that has focus. Movie clips are also included if their `tabEnabled` property is set to `true`, or if they have event handlers associated with them and their `tabEnabled` property is not set to `false`.

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 and movie clips, the focus rectangle's bounding box is determined by the object's *hit area*—the invisible region that (in Flash desktop applications) defines the part of the button or movie clip 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 customize the color of the focus rectangle or disable it. For more information, see "Customizing the focus rectangle" on page 20.

Guidelines for using default navigation

The following are guidelines and considerations for using default navigation in your Flash Lite applications.

- If you disable the default focus rectangle by setting `_focusRect` to `false`, be sure to provide an alternative focus indicator for your buttons, input text fields, and tab-enabled movie clips. For buttons, you can do this by adding a visual element to the button's "over" state—the part of a button object's timeline that's displayed when the button has focus. For an example of this technique, see the sample application in "Creating a simple menu using buttons and default navigation" on page 26. For input text fields, you can use the Selection object to determine when the text field has received focus and display the desired focus indicator. For an example, see the sample application discussed in "Controlling focus with ActionScript" on page 21.
- Have at least two objects (input text fields, buttons, tab-enabled movie clips, or a combination) on the screen at the same time. If the screen contains only one input text field, button, or tab-enabled movie clip, 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 and button events" on page 23.

- When appropriate, consider using the `Selection.setFocus()` method to set the initial focus to a specific object on the screen. This can help guide the user through the interface and reduce the amount of key navigation they must perform. For example, suppose that a screen in your application contains an input text field. Normally, for the user to enter a value in the text field, they would first press a navigation key to give the text field focus, and then press the select key to open the text input dialog box. You could use the following ActionScript to automatically set the keypad focus to the input text field:

  ```
  Selection.setFocus(inputTxt);
  ```

  For more information about controlling focus with ActionScript, see "Controlling focus with ActionScript" on page 21.
- The alignment of objects on the screen is important with default navigation. When objects are offset from each other on the screen, the sequence in which they get focus may not be intuitive for your users. (You can prevent this by making objects consistent in size and aligning them vertically and horizontally as much as possible. With two-way navigation, you can also control the sequence using the `tabIndex` property; for more information, see "About controlling tab order in two-way navigation" on page 22.

Customizing the focus rectangle

The focus rectangle is a default yellow highlight that indicates which button or input text box is currently selected. Movie clips are also included if their `tabEnabled` property is set to `true`, or if they have event handlers associated with them and their `tabEnabled` property is not set to `false`. For more information, see "About the focus rectangle" on page 18.

You can disable the default focus rectangle behavior by setting the global `_focusRect` property to `false`. You can also disable the focus rectangle for specific buttons or movie clips (see `_focusrect (Button._focusrect property)` and `_focusrect (MovieClip._focusrect property)`) in the *Flash Lite 2.x ActionScript Language Reference.*

You can also change the color of the focus rectangle from the default yellow to any other color. To do this you use the `SetFocusRectColor` command, which takes RGB values as parameters. For example, the following code changes the color of the focus rectangle to red:

```
fscommand2("SetFocusRectColor", 255, 0, 0);
```

Controlling focus with ActionScript

You can use the Selection ActionScript object to get and set the current keypad focus, or to be notified when an object receives or loses keypad focus. This is useful, for example, if you want to automatically set the focus to a specific button when your application first loads. Or you may want to be notified when a specific object on the screen has received (or lost) keypad focus so that you can update the screen accordingly.

For example, the following code uses the `Selection.setFocus()` method to set focus to the button instance named `login_btn`:

```
Selection.setFocus(login_btn);
```

The `Selection.onSetFocus` event listener lets you determine when the keypad focus has changed. You can use this event listener, for example, to create a custom focus manager for input text fields, rather than use the default focus rectangle. The following procedure shows how to create a custom focus manager that changes the border color of the TextField object with focus. For a sample of the completed application (custom_focus_manager.fla), see the Flash Lite Samples and Tutorials page at www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample.

To create a custom text input focus manager:

1. Create a new document from the Flash Lite 2.0 template that you created in "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*, and save it as custom_focus_manager.fla.
2. Using the Text tool, create a text field on the Stage.
3. With the text field still selected, in the Property inspector, select Input Text from the Text Type pop-up menu, type **inputTxt_1** in the Instance Name text box, and select the Show Border Around Text option.
4. In the same manner, create another input text field below the first one with the instance name of **inputTxt_2** and select the Show Border Around Text option for the second text field.
5. In the Timeline, select Frame 1 in the layer named ActionScript.
6. Open the Actions panel (Window > Actions) and enter (or copy and paste) the following code:

```
// Disable focus rect globally:
_focusrect = false;
// Create Selection listener object:
var focusListener:Object = new Object ();
```

```
// Define onSetFocus method:
focusListener.onSetFocus = function (oldFocus, newFocus) {
  // Enable/disable selection indicator:
  if (newFocus instanceof TextField) {
    // Set border color of text field with new focus to red:
    newFocus.borderColor = 0xFF0000;
  }
  if (oldFocus != undefined && oldFocus instanceof TextField) {
    // Set border color of text field with old focus to black:
    oldFocus.borderColor = 0x000000;
  }
};
// Add listener to Selection object:
Selection.addListener (focusListener);
// Set initial focus when application loads:
Selection.setFocus (inputTxt_1);
// Enable full-screen mode:
fscommand2 ("FullScreen", true);
```

7. Save your changes and test the application in the emulator (Control > Test Movie).
8. Press the emulator's down and up arrow keys to switch keypad focus between the two text fields. The text field with focus should have a red border, and the text field without focus should have a black border. Press the select key when a text field has focus to make the text input dialog box appear.

About controlling tab order in two-way navigation

Two-way navigation in Flash Lite is analogous to tab navigation in Flash, and therefore supports the `tabIndex` property that allows you to specifically set the tab order of buttons, movie clips, and input text fields. On devices that support four-way navigation, the `tabIndex` property is not supported, so it's not possible to set tab order using the `tabIndex` property for four-way navigation.

To control tab order in two-way navigation, you assign each object's `tabIndex` property a number that specifies the object's order in the default navigation. For example, suppose that an application contains a button (`my_button`), a movie clip (`my_movieclip`), and an input text field (`my_inputTxt`). The following code establishes the tab order so that the button gets focus first, then the movie clip, and finally the input text field.

```
my_button.tabIndex = 1;
my_movieclip.tabEnabled = true;
my_movieclip.tabIndex = 2;
my_inputTxt.tabIndex = 3;
```

Handling key and button events

Event handlers and event listeners specify how the application will respond to user- and system-generated occurrences. For example, when a button has focus and the user presses the select key, an `onPress` event is generated. In addition to using default navigation and responding to related events, a Flash Lite application can listen for and respond to keypress events.

Not all devices and content types support all device keys. For example, on a device that supports two-way navigation (see "Default navigation modes" on page 16) Flash Lite doesn't generate keypress events for the left and right arrow keys. For a list of keys and description of their availability, see "Keys supported by Flash Lite" on page 14.

This section contains the following topics:

Handling button events

You can use buttons to quickly add interactivity to your Flash Lite applications. Flash Lite supports the same button events as Flash Player on desktop computers, although some events (for example, `onDragOut`) are only available on devices that have a mouse or stylus interface. On devices that have a keypad interface only, a button must have keypad focus before it will generate any events.

Flash Lite supports the following ActionScript button events:

Button event	Description
`onDragOut`	Supported only on devices that have a mouse or stylus. Invoked when the user presses the mouse button over the button and the pointer is then dragged outside of the button.
`onDragOver`	Supported only on devices that have a mouse or stylus. Invoked when the user presses and drags the mouse button outside and then over the button.
`onKeyDown`	Invoked when the button has focus and a key is released.
`onKeyUp`	Invoked when the button has focus and a key is pressed.
`onKillFocus`	Invoked when focus is removed from a button.

Button event	Description
`onPress`	Invoked when the user presses the select key on the device when the button has focus.
`onRelease`	Invoked when the user releases the select key on the device when the button has focus.
`onReleaseOutside`	Invoked when the mouse button is released while the pointer is outside the button after the button is pressed while the pointer is inside the button.
`onRollOut`	Invoked when a button loses focus.
`onRollOver`	Invoked when a button receives focus.
`onSetFocus`	Invoked when a button has input focus and a key is released.

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 default navigation" on page 26.

To create a button event handler:

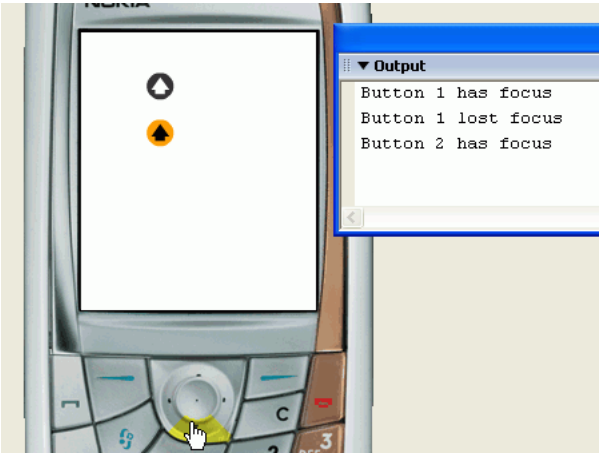
1. Create a new document from the Flash Lite 2.0 template that you created in "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*, and save it as custom_focus_manager.fla.
2. Select Window > Common Libraries > Buttons to open an external library of prebuilt button symbols.
3. In the Library panel, double-click the classic buttons folder to open it, and then open the Circle Buttons folder.
4. Drag an instance of the Menu button symbol to the Stage.
5. In the Property inspector, in the Instance Name text box, type **btn_1**.
6. Drag another instance of the same button to the Stage and position it directly below the first button.
7. In the Property inspector, in the Instance Name text box, type **btn_2**.
8. In the Timeline, select Frame 1 in the layer named ActionScript.
9. Open the Actions panel (Window > Actions) and enter the following code:

```
// Disable the focus rectangle because buttons have an over state
_focusRect = false;

// Event handlers for btn_1
btn_1.onPress = function() {
  trace("You pressed Button 1");
}
```

```
btn_1.onRelease = function() {
  trace("You released Button 1");
}
btn_1.onRollOver = function() {
  trace("Button 1 has focus");
}
btn_1.onRollOut = function() {
  trace("Button 1 lost focus");
}

// Event handlers for btn_2
btn_2.onPress = function() {
  trace("You pressed Button 2");
}
btn_2.onRelease = function() {
  trace("You released Button 2");
}
btn_2.onRollOver = function() {
  trace("Button 2 has focus");
}
btn_2.onRollOut = function() {
  trace("Button 2 lost focus");
}
```

10. 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.



Other types of objects support different events; for example, the `TextField` object includes an `onChanged` event that is invoked when the content of a text field changes. You can write event handler code for these events using the same format as the button event handlers in this procedure. For more information about the events supported for text fields and movie clips, see the `TextField` and `MovieClip` entries in the *Flash Lite 2.x ActionScript Language Reference*.

Creating a simple menu using buttons and default navigation

This section shows you how to create a simple menu using buttons and default navigation. To create the menu, you'll use three button symbols, one for each menu option. Then you'll write event handling code 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" on page 23.

Start with a partially completed Flash document. You can change these settings to target a different device and content type (see "Using the emulator" on page 116).

To create a simple menu using buttons:

1. Download and open the file named simple_menu_start.fla located at www.adobe.com/go/learn_flt_samples_and_tutorials. On the Samples and Tutorials page, locate, download and decompress the .zip file for your Flash Lite version, and then navigate to the Samples folder to access the sample.
2. Open the Library panel (Window > Library).

   Notice that the Library contains three button 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. In the Property inspector, in the Instance Name text box, type **btn_news**.
6. Repeat steps 4 and 5 for the Sports and Weather buttons, naming them **btn_sports** and **btn_weather**, respectively.

7. Align the three buttons vertically, as shown in the following example:

8. In the Tools panel, 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.

9. 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 **txt_status** in the Instance Name text box.

   The Stage should look something like the following image:

10. In the Timeline, select Frame 1 in the layer named ActionScript.

11. Open the Actions panel (Window > Actions) and enter the following code:

```
// Disable the focus rectangle because buttons have an over state
_focusRect = false;

btn_news.onRollOver = function() {
  txt_status.text = "Press to select News"
}
```

```
btn_news.onPress = function() {
  txt_status.text = "You selected News"
}
btn_sports.onRollOver = function() {
  txt_status.text = "Press to select Sports";
}
btn_sports.onPress = function() {
  txt_status.text = "You selected Sports";
}
btn_weather.onRollOver = function() {
  txt_status.text = "Press to select Weather";
}
btn_weather.onPress = function() {
  txt_status.text = "You selected Weather";
}
```

12. 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).

Handling key events

Flash Lite generates key press events in response to the user pressing keys on their device. You can write key handler code to respond to these events For a list of the keys that Flash Lite supports, see "Keys supported by Flash Lite" on page 14.

The following table lists commonly used device keys and the corresponding ActionScript key codes and key code constants for those keys:

Device key	ActionScript key code/key code constant
Select key	Key.ENTER
Up navigation key	Key.UP
Down navigation key	Key.DOWN
Left navigation key	Key.LEFT
Right navigation key	Key.RIGHT
Left soft key	ExtendedKey.SOFT1 (or soft1)
Right soft key	ExtendedKey.SOFT2 (or soft2)
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57
*	56
#	51

Writing an event listener

Event listeners let an object, called a listener object, receive events broadcast by another object, called a broadcaster object. The broadcaster object registers the listener object to receive events generated by the broadcaster. For more information, see "Using event listeners" in *Learning ActionScript 2.x in Flash*.

A easy way to handle key press events is to create a key listener object that defines an `onKeyDown` or `onKeyUp` function, and then register that object with the `Key.addListener()` method. The following example code defines a key listener that responds when the user presses the right navigation key on the device:

```
var myListener:Object = new Object();
myListener.onKeyDown = function() {
  if (Key.getCode() == Key.RIGHT) {
    trace("You pressed the right arrow key");
  }
}
Key.addListener(myListener);
```

For more examples of using key listeners, see the remaining topics in this chapter.

Using a key listener to handle keypress events

The following procedures demonstrate how to use a key listener to handle keypress events in a simple application. The application uses the four navigation keys to move a movie clip around the Stage.

To use an event listener to handle keypress events:

1. Create a new document from the Flash Lite 2.0 template that you created in "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*, and save it as keylistener.fla.
2. Select the layer in the Timeline named Content.
3. Using the Oval tool, create an 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. In the Timeline, select the first frame in Layer 1.

6. Open the Actions panel (Window > Actions), and enter the following code:

```
var myListener:Object = new Object();
myListener.onKeyDown = function() {
  if (Key.getCode() == Key.LEFT) {
    circle._x -= 10;
  } else if (Key.getCode() == Key.RIGHT) {
    circle._x += 10;
  } else if (Key.getCode() == Key.UP) {
    circle._y -= 10;
  } else if (Key.getCode() == Key.DOWN) {
    circle._y += 10;
  }
};
Key.addListener(myListener);
```

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

   Press the four navigation keys on the emulator's keypad (or the corresponding arrow keys on your keyboard) to make the circle move around the Stage.

Using the soft keys

To use the soft keys in your Flash Lite application, you must first call the `SetSoftKeys` command. Subsequently, Flash Lite generates an `ExtendedKey.SOFT1` event when the user presses the left soft key and an `ExtendedKey.SOFT2` event when the user presses the right 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 example shows the result of using this command in 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 shown in the following example:

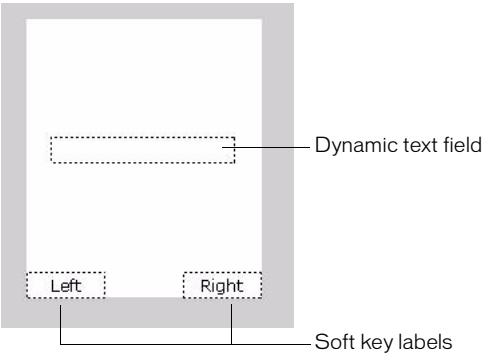
For more information about the `SetSoftKeys` command, see the `fscommand2 function` entry in the *Flash Lite 2.x ActionScript Language Reference*.

To use the soft keys in an application:

1. Create a new document from the Flash Lite 2.0 template that you created in "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*, and save it as softkey.fla.
2. Using the Text tool, create a static text field named Left and position it in the lower-left corner of the Stage, above the left soft key on the device.

3. 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.
4. Using the Text tool, create a dynamic text field, and position it in the middle of the Stage.

   Your document's Stage should look like the following example:



5. With the dynamic text field still selected, in the Property inspector, type **status** in the Instance Name text box.
6. Open the Actions panel (Window > Actions) and, in the Timeline, select Frame 1 in Layer 1.
7. In the Actions panel, type the following code:

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

8. Create and register an object to respond to keypress events (see "Using a key listener to handle keypress events" on page 31) by entering the following code in the Actions panel:

```
var myListener:Object = new Object();
myListener.onKeyDown = function() {
  if (Key.getCode() == ExtendedKey.SOFT1) {
    // Handle left soft keypress event.
    status.text = "You pressed the Left soft key.";

  } else if (Key.getCode() == ExtendedKey.SOFT2) {
    // Handle right soft keypress event.
    status.text = "You pressed the Right soft key.";
  }
};
Key.addListener(myListener);
```

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

   To test the application, 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.

CHAPTER 3‘

Working with Text and Fonts 3

This chapter describes how you can add static and dynamic text fields and add input text fields to your Macromedia Flash Lite 2.x from Adobe applications.

This chapter contains the following topics:

About text in Flash Lite

Flash Lite 2.x supports the following text features:

- Static, dynamic, and input text fields

  At run time, the contents of a static text field can’t change, but the contents of a dynamic or input text field can change. Input text fields allow users to enter text. Flash Lite 2.1 supports inline text input on most devices. On devices that support complex languages and in Flash Lite 2.0, input text fields use the device’s generic text input mechanism. For more information about input text fields, see “Using input text fields” on page 39.

- Embedded and device fonts

  You can have Flash Lite render text fields using font outlines that you embed in the SWF file or using fonts available on the device. For more information about font rendering methods, see “Font rendering methods in Flash Lite” on page 47.

- Unicode text encoding

  Flash Lite can display text in any language as long as fonts containing the required character glyphs are available. For information on multilanguage authoring in Flash, see "Creating Multilanguage Text" in *Using Flash*.
- Partial support for HTML formatting and the TextFormat class properties
- Scrolling text

Flash Lite does not support all the text features that are in the desktop version of Flash Player. Flash Lite has the following limitations:

- Advanced anti-aliasing, the enhanced font rendering technology available in Macromedia Flash Player 8 from Adobe and later, is not supported.
- Text formatting is supported, but the following limitations apply for device text:
  - Only the color, face, size, bold, and italic options are available.
  - Formatting is not displayed if the device text font does not include the selected option. For example, a field formatted as italic appears as regular text when the device font does not include an italic version.
- Device text cannot be masked, used as a mask, or rendered with transparency.
- The Render Text as HTML formatting option is partially supported for input and dynamic text fields. Text is displayed without visible HTML tags, but formatting is honored only for the following tags: `p`, `br`, `sbr`, `font` (with `face`, `color`, and `size` attributes), `b`, and `i`.
- Flash Lite does not support Cascading Style Sheets (CSS).
- Flash components, including Label, TextArea, and TextInput, are not supported.
- The TextField and TextFormat objects are partially supported, and additional limitations apply for Arabic, Hebrew, and Thai. For more information, see the *Flash Lite 2.x ActionScript Language Reference*.
- The XML and XMLNode objects are supported.

Flash Lite 2.1 adds support for inline text input, predictive text engines, and XMLSocket:

- Inline text input support lets the user enter text directly into text fields.
- Predictive text support enables functionality such as word completion and candidate lists. Flash Lite 2.1 supports the top predictive text engines (such as T9, eZiTap/eZiText, and iTap) on any platform, as long as they are implemented similarly to the standard APIs provided by the predictive text engine vendors.
- XMLSocket support extends the Flash desktop support to Flash Lite, and enables developers to create continuous, low-latency data connections for applications such as games and chat.

Creating and formatting text

You create and format text in Flash Lite as you would for a Flash desktop application.

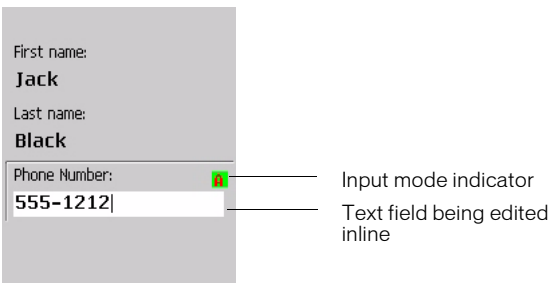
For more information about working with text in Flash, see the following topics in *Using Flash*:

- "Creating text"
- "Setting text attributes"
- "Editing text"
- "Controlling text with ActionScript"

For a list of text features that are not supported in Flash Lite, see "About text in Flash Lite" on page 33.

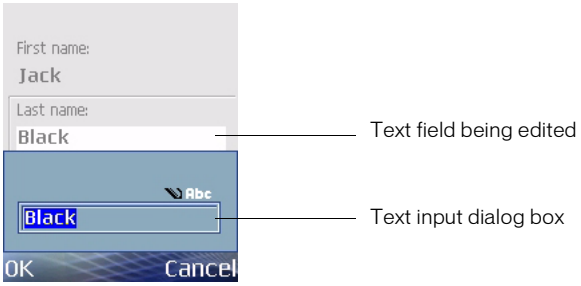
Using input text fields

Flash Lite 2.1 supports inline text input. This features lets users edit text fields directly in the Flash Lite application, rather than in a separate text input box as in earlier versions. For example, the following figure shows how an inline input text field appears on a Series 60 device from Nokia:



For more information about inline text input, see "Using inline text input (Flash Lite 2.1)" on page 36.

On devices running Flash Lite 2.0 and Flash Lite 1.1, and in the Adobe Device Central emulator for all versions, users edit the contents of input text fields using a modal dialog box that appears over the Flash Lite content. For example, the following figure shows an example of a text input dialog box on a Symbian Series 60 device running Flash Lite 2.0:



In general, existing Flash Lite 2.0 and Flash Lite 1.1 applications can work without modification in Flash Lite 2.1. In place of the modal dialog box, users can simply edit input text fields inline. However, legacy content should be re-authored to use the features that are available only in Flash Lite 2.1, such as the ability to set text selections or the `activateTextField` command.

For more information about the input text dialog box, see "Using the device's input text dialog box (Flash Lite 2.0)" on page 40.

Using inline text input (Flash Lite 2.1)

Flash Lite 2.1 supports inline text input, which allows users to enter and edit text directly in an input text field. While entering text in a Flash Lite application, users interact with the device's native input method editor (IME)—the Flash Lite player does not process the user input itself.

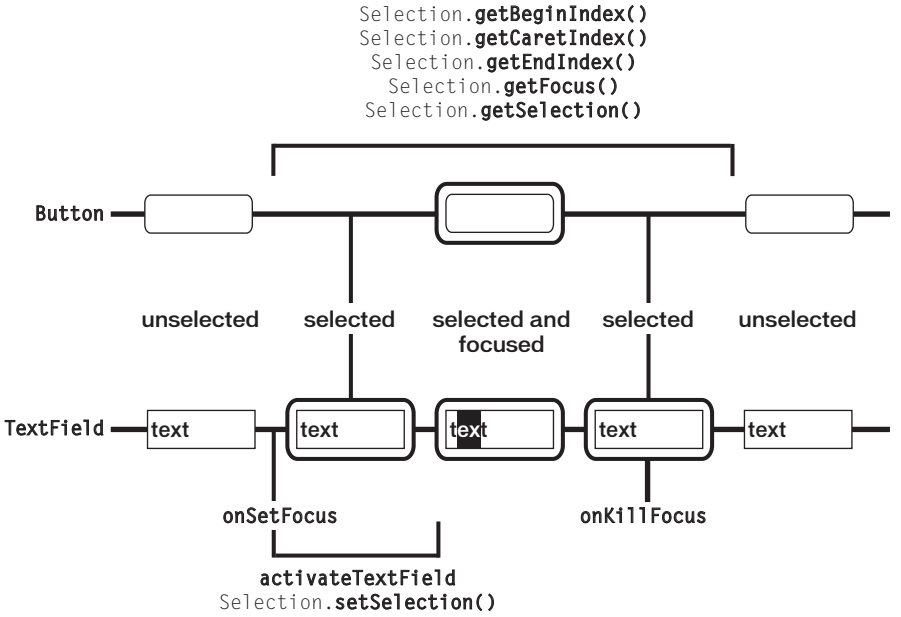
NOTE

The Flash Lite emulator on Adobe Device Central does not show inline text. For more information, see "Testing inline text in Flash Lite 2.1" on page 122.

While an input text field is active, the Flash Lite 2.1 player runs in *restricted* mode. In this mode, the device's native IME handles all key press events, and no ActionScript is processed. In addition, all animation, sound, and video stop playing events are ignored. After a text field is deactivated, Flash Lite resumes normal operating mode.

Before a text field can accept text input from a user, it must be *activated*. A user can activate an input text field by giving it selection focus and then either pressing the device's Select key or pressing a soft key for which the application sets the text entry to the focused state. (For details about activating a text field, see "Activating input text fields with ActionScript" on page 38.)

The following figure shows the different states of an input text field: not selected, selected (but not activated), and activated:



About inline input text fields and navigation

An activated text field contains a pointer that indicates the current insertion point. The user can change the pointer's location within the text field using the device's navigation keys.

By default, Flash Lite draws a focus rectangle around the text field that has the current focus. This focus rectangle can sometimes obscure an activated text field's flashing pointer or insertion indicator. For this reason, it's recommended that you disable the focus rectangle (by setting `_focusRect` to `false`) and use a custom focus indicator. For an example of an application that uses a custom focus indicator, see the Flash Lite Samples and Tutorials page at www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample.

Activating input text fields with ActionScript

A user can activate an input text field that has keypad focus by pressing the device's Select key. A Flash Lite application can also automatically activate an input text field when it receives focus using the `activateTextField` command. This command activates the currently selected text field; if there is no selected field when the command is executed, nothing happens.

The most common place to call `activateTextField` is from within a `Selection.onSetFocus` handler or a `TextField.onSetFocus` handler. For example, suppose that your application contains two (or more) input text fields on the Stage. The following code automatically activates the text field that receives focus:

```
var focusListener:Object = new Object ();
focusListener.onSetFocus = function (oldFocus, newFocus) {
  // Call activateTextField:
  fscommand ("activateTextField", "");
};
TextField1.addListener (focusListener);
```

You can also use the `TextField.prototype.onSetFocus` handler to activate all text fields whenever they receive focus.

It is also possible to use keys other than the device's Select key to trigger the `activateTextfield` command. The following code activates a text field for all number keys, which makes it easier to enter (for example) the letter "a". For example, if the application includes `activateTextfield` in the `TextField.onSetFocus` handler, the user would have to press Select and then 2; this code allows the user press `2` twice, which is more intuitive.

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  if (Key.getCode() >= 48 && Key.getCode() <= 57 ){
    fscommand("activateTextField", "");
  }
Selection.addListener (keyListener);

};
```

For a completed sample application that uses this technique, see the Inline Text Input sample at www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample.

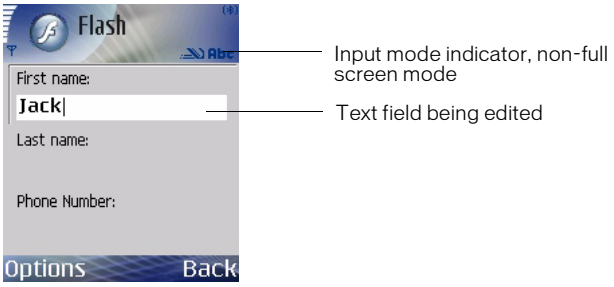
Supported languages

Flash Lite 2.1 supports inline text input for Latin and Asian languages but not complex languages, which include right-to-left languages such as Arabic, Hebrew, Urdu, Farsi, and Yiddish, as well as some Asian languages.

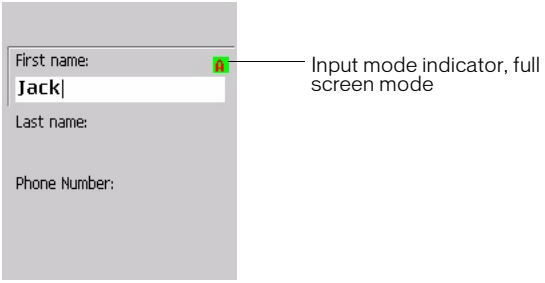
About the input mode indicator

To assist users with common text input tasks, most devices provide several input modes, such as predictive or manual (*triple tap* or *multi tap*) text entry, or all number-only modes.

When Flash Lite is running in full screen mode, the device displays a letter A for alpha input mode and a number sign (#) for numeric input mode. When Flash Lite is not running in full screen mode, the device could draw an input mode indicator on a status bar or some other location on the display. For example, the following image shows the input mode indicator in non-full screen mode on the Series 60 stand-alone version of Flash Lite 2.1:



When the player is running in full-screen mode, the device may draw an input mode indicator to a location of its choice on the display. For example, the following figure shows the input mode indicator in full screen mode on the Series 60 stand-alone version of Flash Lite 2.1:



The input mode indicator for full screen applications shown in the previous figure is a reference implementation for the stand-alone Series 60 player. The specific indicator, if any, that appears is determined by the device.

Using the device's input text dialog box (Flash Lite 2.0)

To open the device's input dialog box, users must first give an input text field focus, and then press their device's Select key.

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 selects 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 selects Cancel (the right soft key), no text is assigned to the input text field.

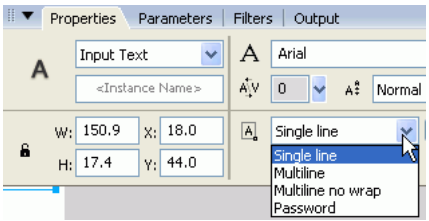
The Adobe Device Central emulator mimics the features of the text input dialog box when you test your application in the Flash authoring tool. The following image shows the text input dialog box running in the emulator:

Text input dialog box running in the emulator

For an example of using an input text field in an application, see "Text field example application" on page 47.

Specifying 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 shows a single line input text box. The input text box scrolls horizontally if the user enters more characters than can be displayed.

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 users to provide a numeric value, such as their age, and that the input text field has the variable name `ageVar`. To ensure that the user enters only numeric values in the text input dialog box, you could add the following code to your application:

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

When users open the input text dialog box, they can enter only numeric values in the text fields.

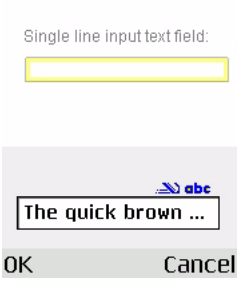
For more information, see `SetInputTextType` in the `fscommand2 function` entry in the *Flash Lite 2.x ActionScript Language Reference*.

Input text dialog boxes (Flash Lite 2.0)

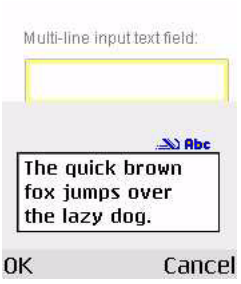
In Flash Lite 2.0, the user enters text into a separate modal dialog box (controlled by the host application, not Flash Lite) rather than interacting with the on-screen text field directly. During this interaction, the Flash Lite player is effectively paused until the user exits the dialog box.

(In Flash Lite 2.1, the user can enter text directly into the on-screen text field.)

The following image shows a device's input text dialog box for a single line input text field in a Flash Lite 2.0 application:



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 displays each character that the user enters. When the user clicks OK, the entire password is masked with asterisks, as the following image shows:

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 at run time (for example, _sans is mapped to a sans serif font, if available).

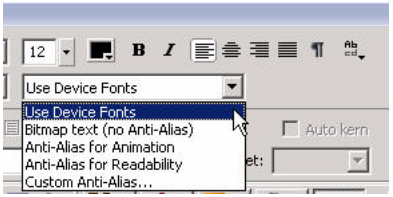
Render the font as a bitmap Flash Lite renders bitmap text by aligning font outlines to pixel boundaries, which makes text readable at small point sizes (such as 10 points or smaller). This option requires that you include font outlines in the published SWF file for the selected font. (See "Embedding font outlines in SWF files" on page 50.)

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. (See "Embedding font outlines in SWF files" on page 50)

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 developers. The other two methods (Anti-Alias for Readability and Custom Anti-Alias) are available only to applications that are targeting Flash Player 8 or later on desktop computers.

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. (See "Embedding font outlines in SWF files" on page 50.)
   - Select Anti-Alias for Animation to have Flash Lite anti-alias the text field's font according to the current rendering quality setting (see "Flash Lite rendering quality and anti-aliased text" on page 49). This option requires that Flash embed font outlines in the published SWF file.

Below are some guidelines to consider when you choose between anti-aliased, bitmap, and device text:

- If you're using embedded fonts with dynamic or input text fields, embed the outlines only for the characters that you need to display. This will help to reduce file size. For example, if you're using an input text field to capture a user's age (a number), include only the font outline for number characters (0-9). In this case, consider restricting the character input to numbers (see "Font rendering methods in Flash Lite" on page 47).
- The Adobe Device Central emulator does not emulate device fonts, unless you have the same fonts installed on the computer that you're using to develop the content. Therefore, the layout and appearance of your device text field might be different in the emulator than on the device.
- If you apply one of the generic device font faces (_sans, _serif, or _typewriter), Flash Lite tries to find a similar font on the device to display the text. However, because mobile devices typically have fewer fonts and font styles than a desktop computer, a font such as _sans might not map to a sans serif font. You must test the application on each target device to determine the proper font settings.

- Anti-aliased text in Flash Lite is, essentially, a complex vector shape. Like any vector shape, it takes some processing power to render. Because processing speed on most devices is relatively slow, animating a lot of anti-aliased text may degrade application performance. To improve performance, try temporarily lowering the Flash Lite player's rendering quality during the animation, and then returning it to the higher rendering quality when the animation is complete. For more information, see the next section.

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 renders outlines using medium quality. You can control the rendering quality using the `SetQuality` command (see `SetInputTextType` in the `fscommand2 function` entry in the *Flash Lite 2.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 figure shows 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 (see "Font rendering methods in Flash Lite" on page 47). 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.

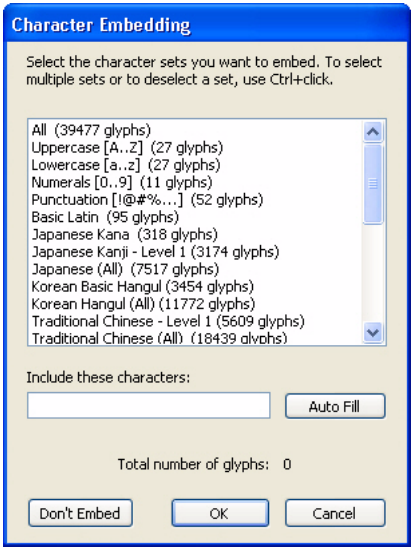
For static text fields that use the anti-alias or bitmap font rendering methods, Flash automatically embeds the font outlines required to display the text field's contents. For example, if a static text field contains the word *Submit*, Flash automatically embeds the font outlines required to display those six characters (*S*, *u*, *b*, *m*, *i*, and *t*). Because the contents of a static text field can't change, the SWF file needs to include font outlines only for those 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 or 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 you want to embed from the list, type the characters that you want to embed in the text box, or click Auto Fill to include the characters that are in the selected text field.
5. Click OK.

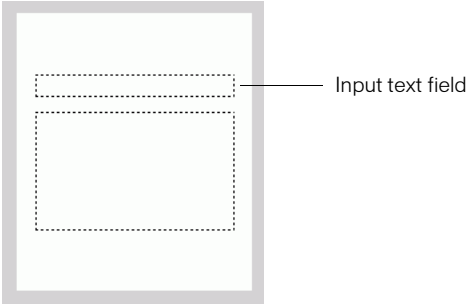
Text field example application

This section describes how to create a simple application that gets text input from the user, and then formats and displays that text in an HTML-enabled dynamic text field. The application also uses the `SetFocusRectColor` command to change the focus rectangle color from the default color (yellow) to black.

For a completed sample application (textfield_example.fla) that uses this technique, see www.adobe.com/go/learn_flt_samples_and_tutorials. Locate the .zip file for your version of ActionScript, download and decompress the .zip file, and then navigate to the Samples folder to access the sample.

To create the text field example application:

1. In Flash, create a new document from the Flash Lite 2.0 template that you created earlier in "Creating a Flash Lite document template (Flash Professional only)" in *Getting Started with Flash Lite 2.x*, and save it as textfield.fla.
2. Using the Text tool in the Tools panel, create a single-line text field across the top of the Stage.
3. With the text field still selected, 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, and type **inputTxt** in the Instance Name text box.
4. Create another text field below the first that is several times taller than the first one, as shown below:



5. With the second text field selected, in the Property inspector, select Dynamic Text from the Text Type pop-up menu, select Multiline from the Line Type pop-up menu, select the Render Text as HTML option, select Use Device Fonts from the Font Rendering Method pop-up menu, and type **messageTxt** in the Instance Name text box.
6. In the Timeline, select Frame 1 on Layer 1.

7. Open the Actions panel (Window > Actions) and enter the following code:

```
Selection.setFocus(inputTxt);
fscommand2("SetFocusRectColor", 0, 0, 0);
inputTxt.onChanged = function() {
  messageTxt.htmlText = "You entered: <i>" + this.text + "</i>";
}
```

   The `Selection.setFocus()` method sets the initial focus to the input text field (`inputTxt`). Next, the `fscommand2()` function call specifies a custom focus rectangle color. Last, the input text field's `onChanged` event handler, called whenever the contents of the input text field changes, formats and displays the text that the user entered in the `messageTxt` text field.

8. Save your changes and start the application in the emulator (Control > Test Movie).
9. To use the application, press the emulator's Select key to open the text input dialog box and enter some text using your computer's keyboard. Then click OK to close the dialog box. The text that you entered appears in the `messageTxt` text field in italics.

Creating scrolling text

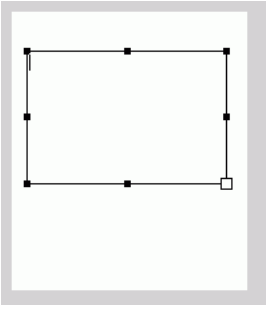
Flash Lite supports the `TextField.scroll` and `TextField.maxscroll` properties, which let you create scrolling text fields. The `scroll` property specifies the first visible line in a text block; you can get and set its value. For example, the following code scrolls the text field whose variable name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

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

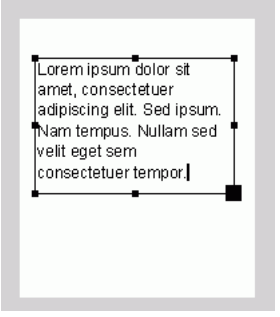
The `maxscroll` property specifies the first visible line in a text block when the last line of the text is visible in the text block; this property is read-only. You can compare a text field's `maxscroll` property to its `scroll` property to determine how far a user has scrolled within a text field. 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 2.0 template that you created earlier in "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*.
2. Using the Text tool, 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 Instance Name text box in the Property inspector.
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. In the Timeline, select the first frame on Layer 1, and open the Actions panel (Window > Actions).

10. Enter the following code in the Actions panel:

```
var keyListener:Object = new Object();
keyListener.onKeyDown = function() {
  var keyCode = Key.getCode();
  if (keyCode == Key.DOWN) {
    story.scroll++;
  }
  else if (keyCode == Key.UP) {
    story.scroll--;
  }
};
Key.addListener(keyListener);
```

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

    Click the up and down navigation keys on the emulator (or the Up Arrow and Down Arrow keys your computer's keyboard) to scroll the text up or down.

CHAPTER 4

Working with Sound, Video, and Images

4

This chapter describes how to incorporate sound, video, and external images in a Macromedia Flash Lite 2.x from Adobe application.

This chapter contains the following topics:

About sound in Flash Lite

Flash Lite 2.0 supports two types of sound: standard (*native*) Flash sound and device sound. Native sounds are played directly by the Flash Lite player, just as they are in the desktop version of Flash Player. With some exceptions, you can use native sounds in your Flash Lite application as you would use them in a Flash desktop application. For more information about using native Flash sounds, see "Using native Flash sounds" on page 65.

Device sounds, in contrast, are played directly by the device, rather than by the Flash Lite player. Some examples of device sound formats include MIDI or MFi, but supported sound formats vary from device to device. To use device sounds, you can either bundle them in the published SWF file, or you can load external sound files from a network location or the device's local file system. Device sounds have some limitations that don't apply to native sounds. For more information about using device sounds, see "Using device sound" on page 58.

Using device sound

Device sounds refer to audio that is played directly by the device, rather than by the Flash Lite player. Different devices may support different sound formats, including MIDI, MFi, or MP3. To incorporate device sounds in your Flash Lite content you can either include them within your published SWF file, or load external sound files over the network or from the device's local file system. With some exceptions, you can use the ActionScript Sound object to control device sounds, as you would control sounds in the desktop version of Flash Player.

This section contains the following topics:

Using bundled device sound

To bundle a device sound in your application, you first import a proxy sound in a format that the Flash authoring tool recognizes, such as an MP3, WAV, or AIFF file. Then you link the proxy sound to a device sound file on your computer that you want to bundle in your application. During the SWF file publishing process, the Flash authoring tool replaces the proxy sound with the linked external sound. During playback, Flash Lite passes the sound data to the device to decode and play.

You can also package multiple device sounds in different formats in a single Flash sound bundle (FLS) file. This is useful if you're creating the same content for several devices that support different device sound formats. For more information, see "Creating a sound bundle" on page 61.

This following procedure demonstrates how to import and play a bundled device sound. To play a device sound you can either attach it to the Timeline or use the Sound object to play the device sound with ActionScript. This section explains both techniques.

To import and play a device sound:

1. Create a new document from the Flash Lite 2.0 Symbian Series 60 document template, and save it as **device_sound.fla**.

   For more information about using the Flash Lite document templates, see "Creating a Flash Lite document template" in *Getting Started with Flash Lite 2.x*.

2. Select File > Import > Import to Library. On the Samples and Tutorials page at http://www.adobe.com/go/learn_flt_samples_and_tutorials, locate, download and decompress the .zip file for your Flash Lite version, and then navigate to the Samples folder.

3. Select the proxy.wav file and click OK.

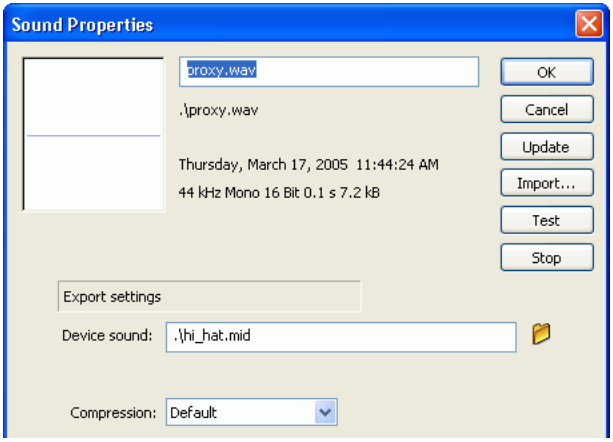
   NOTE: You can use any sound file that's recognized by the Flash authoring tool as the proxy sound. The proxy.wav file is provided for your convenience.

4. To link the proxy sound to the device sound file, do the following:

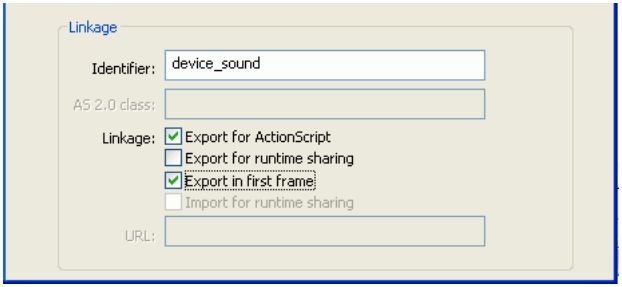
   a. In the Library panel, right-click (Windows) or Control-click (Macintosh) the proxy sound symbol and select Properties from the context menu to open the Sound Properties dialog box.

   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 http://www.adobe.com/go/learn_flt_samples_and_tutorials. On the Samples and Tutorials page, locate, download and decompress the .zip file for your Flash Lite version, and then navigate to the Samples folder and select the file named hi_hat.mid.



d. (Optional) To control the device sound with ActionScript, click Advanced to display the advanced sound properties options, select Export for ActionScript, and type **device_sound** in the Identifier text box.



e. Click OK to close the Sound Properties dialog box.

To play the device sound, you can either attach the proxy sound to the Timeline or use the ActionScript sound object. To use the ActionScript sound object, skip to step 6.

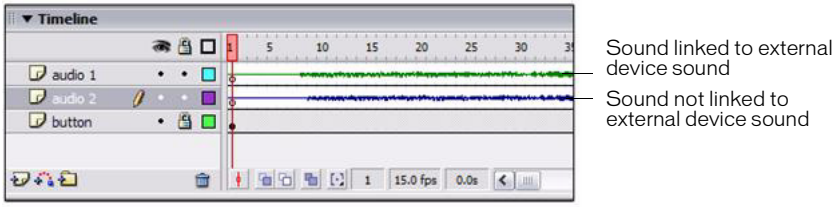
5. To attach the device sound to the Timeline, do the following:

a. Select the keyframe on Frame 1 on the layer named Content.

b. In the Property inspector, select proxy.wav from the Sound pop-up menu.

This attaches the proxy sound to the keyframe.

The Flash authoring tool displays the proxy sound's waveform in the Timeline. 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, as the following image shows.



6. To play the sound with ActionScript, do the following:
   a. Select the layer named Actions in the Timeline.
   b. Open the Actions panel (Window > Actions), and type the following code:

```
var deviceSound:Sound = new Sound();
deviceSound.attachSound("device_sound");
deviceSound.start();
```

7. Select Control > Test Movie to start the Adobe Device Central emulator and test your SWF file.

Creating a sound bundle

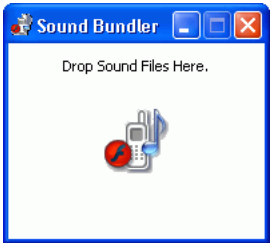
Flash Lite 2.0 provides the ability to encapsulate several device sounds of different formats into a single *sound bundle*. As an example, a single Flash application can contain the same sound represented in both MIDI and MFi formats. When the application is played on a device that supports MIDI audio, Flash Lite selects the MIDI sound data from the sound bundle and passes it to the device to play. Similarly, if the application is played on a device that supports only MFi, Flash Lite passes the MFi sound data to the device.

You use a utility called the Flash Lite Sound Bundler to create a sound bundle (FLS) file. You then link the FLS file to a proxy sound in your Flash Lite document, just as you would do for a single device sound. For more information about adding device sounds to your Flash Lite applications, see "Using bundled device sound" on page 58.

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

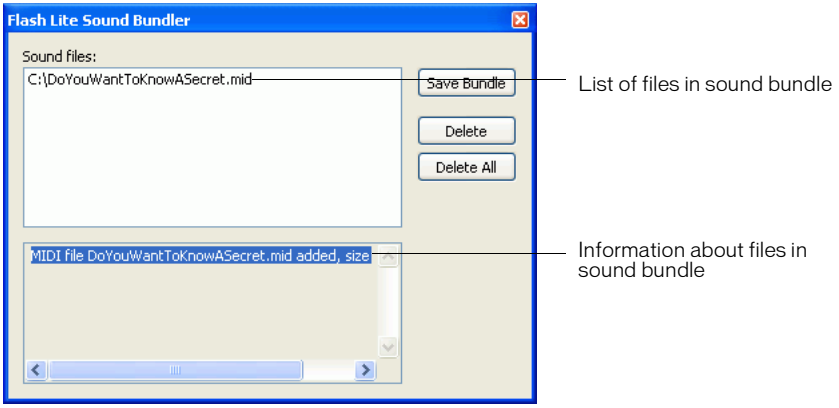
To create a sound bundle file:

1. Open the Flash Lite Sound Bundler application (FlashLiteBundler.exe) located in the Flash installation folder (for example, *boot drive*/Program Files/Adobe/Flash CS3/FlashLiteBundler.exe).



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

   Another window appears that lists the contents of 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 the file in the list and click Delete.

   To delete all files in the sound bundle, click Delete All.

5. To save the sound bundle, click Save Bundle, and choose a name and location for the FLS file.

6. To exit the Sound Bundler, click the close button (X) in the Sound Bundler window.

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" on page 58).

Playing external device sounds

In addition to playing device sounds that are bundled in the published SWF file (see "Using bundled device sound" on page 58), you can also load and play external sound files. To play external device sounds you use the `loadSound()` method of the Sound object. As with bundled device sound, the Flash Lite player passes the externally loaded audio to the device to decode and play.

The following information about playing external device sounds in Flash Lite 2.0 is important to remember:

- Unlike in the desktop version of Flash Player, externally loaded sounds in Flash Lite 2.0 are always treated as event sounds. This means that external device sounds do not stream—that is, play as they are being downloaded. The sound data must download fully before you can play the sound. For the same reason, you must call the Sound object's `start()` method to play the sound once it is fully loaded (see the following code example).
- The Flash Lite 2.0 implementation of the `loadSound()` method does not support that method's second parameter (*`isStreaming`*). Flash Lite ignores this parameter if it is present.
- Flash Lite 2.0 does not play externally loaded MP3 files natively. If your application loads an external MP3 file, Flash Lite passes the sound data to the device to decode and play, as it does with any externally loaded sound file.

The following code shows how to load and play an external sound file:

```
// Create the sound object.
var mySound:Sound = new Sound();
// Define onLoad handler for sound,
// which starts the sound once it has fully loaded.
mySound.onLoad = function(success){
  if(success == true) {
    mySound.start();
  }
}
// Load the sound.
mySound.loadSound("http://www.adobe.com/audio.midi");
```

About synchronizing device sounds with animation

Device sounds in Flash Lite are always treated as event sounds. This means that you can't synchronize device sounds with animation in the timeline in the same manner that you can with native Flash sounds. However, you can use device sounds to approximate true synchronized sound by setting the Flash Lite player's `_forceframerate` property to `true`. When this property is set to `true`, Flash Lite drops frames from animation to maintain the SWF file's specified frame rate. As long as the device sound data is authored for the correct duration, and as long as the device plays back sound data at the expected rate, animation and sound will be close to being synchronized.

For example, suppose you have a device sound that's 5 seconds long. During playback, you'd like this sound to play in synch with animation in the timeline. Also assume that your application's frame rate is set to 15 FPS. When you start the sound—either by attaching it to a frame in the timeline or calling `Sound.start()`—you simultaneously set `_forceframerate = "true"`. Subsequently, for each second of device audio playback, Flash Lite ensures that the playback head has advanced 15 frames in the timeline. If, for some reason, the player cannot render each frame in the animation during this time, it drops frames to maintain the specified frame rate.

For more information about the `_forceframerate` property, see `_forceframerate property` in the *Flash Lite 2.x ActionScript Language Reference*.

Determining supported device sound formats

The `System.capabilities.audioMIMETypes` property contains an array of audio MIME types that the device supports. You can use this information to determine what types of device audio your application can play. The array's indexes are the same as the supported MIME types, so you can easily check whether a device supports a specific type of audio.

For example, the following code first checks whether the device supports playback of MIDI audio before loading an external MIDI file:

```
if (System.capabilities.audioMIMETypes["audio/midi"]) {
  my_sound.loadSound("soundtrack.mid");
}
```

Using native Flash sounds

In addition to supporting device sound, Flash Lite 2.0 also supports standard, or *native*, Flash sounds. Essentially, a native sound is a sound in any format that the Flash authoring tool recognizes. Native sound in Flash Lite can play either as event sound or synchronized sound (see "Using device sound" on page 58).

The general workflow for using native sounds in your Flash Lite applications is the same as for Flash desktop applications, with the following exceptions:

- Flash Lite does not support playing external MP3 files *natively*. However, if the device can play MP3 audio, you can load external MP3 files in your application. For more information, see "Playing external device sounds" on page 63.
- Flash Lite does not support the Speech audio compression option.

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*.
- "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*.

Using the 8kHz sampling rate feature

By default, the Flash authoring tool exports native audio at sampling rates of 5, 11, 22, or 44 kilohertz (kHz). However, many devices don't support playback of audio data at these sampling rates. For example, the Nokia Series 60 devices support only 8 and 16 kHz. In previous versions of Flash Lite, the player resampled the audio at runtime to 8 kHz before calling the device's native sound application programming interfaces (APIs) to play the sound. This resampling required additional processing time and memory consumption.

In Adobe Flash CS3 Professional, you can choose to have the authoring tool resample your application's native audio at 8 kHz during the SWF file publishing process. You apply this setting to all stream and event sounds in your application, or just to specific sounds in your document's Library panel.

The first procedure that follows explains how to enable the 8kHz sampling rate option for individual sounds in the Sound Properties dialog box; the second procedure explains how to use the Publish Settings dialog box to set this option globally for all event and stream sounds.

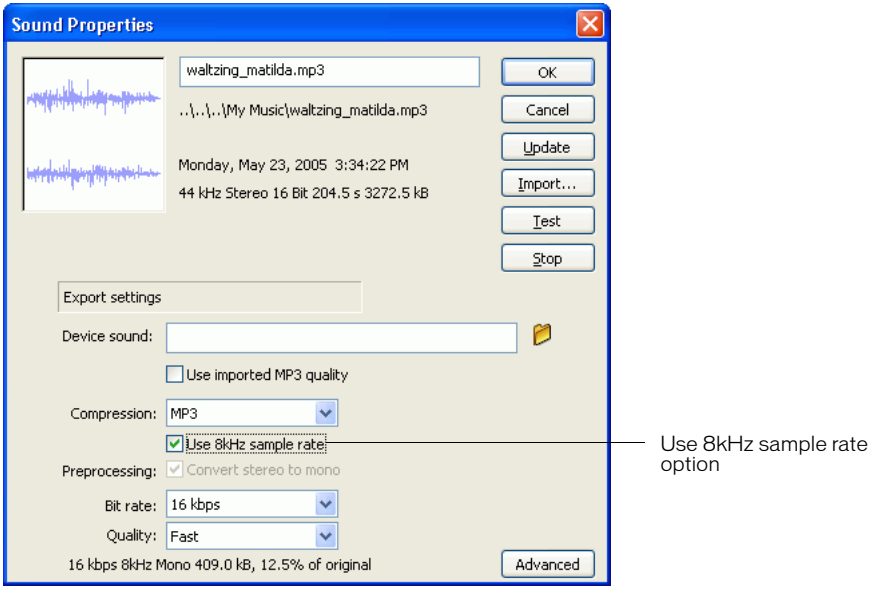
NOTE

For stream sounds, Flash always applies the global sound compression options that you specify in the Publish Settings dialog box. The per-sound compression options that you specify in the Sound Properties dialog box apply only to event sounds.

To enable the 8kHz sampling rate feature for an individual sound:

1. In Flash, right-click (Windows) or Control-click (Macintosh) a sound symbol in the Library panel and choose Properties from the context menu.
2. In the Sound Properties dialog box that appears, select MP3 from the Compression pop-up menu.

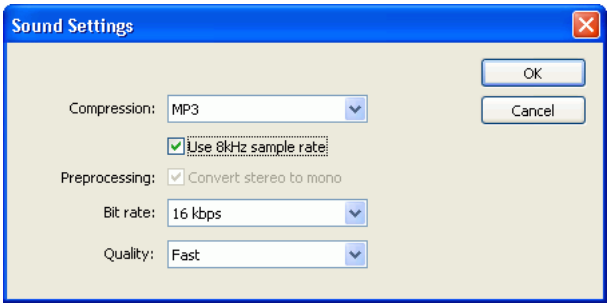
   The 8kHz sampling feature is available only for MP3-compressed audio.
3. Deselect the Use Imported MP3 Quality option, if selected.
4. Select the Use 8kHz Sample Rate option.



5. Click OK.

To enable the 8kHz sampling rate feature globally for all native sounds:

1. In Flash, select File > Publish Settings.
2. In the Publish Settings dialog box, click the Flash tab.
3. To enable the 8kHz sampling rate for all stream sounds in your application, do the following:
   a. Click Set Stream sound options.
   b. In the Sound Properties dialog box that appears, select MP3 from the Compression pop-up menu.
      The 8kHz sampling feature is available only for MP3-compressed audio.
   c. Select the Use 8kHz Sample Rate option.



   d. Click OK to close the Publish Settings dialog box.
4. To enable the 8kHz sampling rate for all event sounds, do the following:
   a. Click Set Event sound options.
   b. In the Sound Properties dialog box that appears, select MP3 from the Compression pop-up menu.

NOTE	The 8kHz sampling feature is available only for MP3-compressed audio.

   c. Select the Use 8kHz Sample Rate option.
   d. Click OK.
5. Click OK to close the Publish Settings dialog box.

Using device video

Flash Lite 2.0 can play *device video*, which refers to any video format or encoding that's supported by the target device. To keep player size small (and to support a wide variety of video formats), Flash Lite 2.0 does not decode or render device video natively. Instead, Flash Lite 2.0 relies on the device to decode and render device video to the device's display. For this reason, there are some limitations to using device video. For more information, see "Limitations of device video" on page 77.

Some common device video formats include 3GP, 3G2 (or 3GPP2), and MPEG-4. In general, you can play any video format in your Flash Lite 2.0 application that the target device supports. Different devices support different video codecs and formats. To determine what video formats a specific device supports, check the device's specifications from the device manufacturer or use ActionScript to query the device's supported formats (see "Determining supported video formats" on page 77).

To deploy device video, you can either bundle it in the SWF file's library (see "Using bundled device video" on page 69) or load it from an external file on the user's device or from a network location (see "Using external device video" on page 75).

To control device video playback, use the ActionScript Video object. Flash Lite 2.0 adds additional methods to the Video object that are not available in the desktop version of Flash Player, including `Video.play()` and `Video.pause()`. For more information about using the Video object to control video playback, see "Controlling video with ActionScript" on page 78.

This section contains the following topics:

Using bundled device video

One way to deploy video in your Flash Lite 2.0 application is to package (bundle) the video files within your published SWF file. To use bundled video, you import a device video into your document's library. When you publish your application, the authoring tool bundles the device video file into the published SWF file.

Bundling device video in a SWF file can add significantly to the size of your published SWF file, but it provides the best portability and playback reliability, because no external files need to open or stream.

This section contains the following topics:

Importing device video

To bundle video in your Flash Lite application, you first import the video into your Flash document. To do this, you can either use the Import Video wizard or the import feature available in the Video Properties dialog box. Both techniques result in a video symbol in your document's library that contains the original device video data. When you publish your application, the authoring tool bundles the external video file's data in the published SWF file.

To import a device video by using the Import Video wizard:

1. In Flash, select File > Import > Import Video.

   The Video Import wizard appears.
2. Select the option to import a file on your computer and click Browse.
3. Browse to the folder that contains the device video file and select it.

   If you don't see the desired video file listed in the Open dialog box (or if you can see it but can't select it), select All Files (*.*) from the Files of Type pop-up menu (Windows), or All Files from the Enable pop-up menu (Macintosh). This may be necessary because the Flash authoring tool cannot recognize most device video formats.
4. Click Open.
5. In the Import Video wizard, click Next to go to the wizard's Deployment screen.

   For Flash Lite 2.0 SWF files, the only deployment option is to bundle the device video in the SWF files.

6. Click Finish to import the video.

   A new video symbol that's linked to the device video file appears in your document's Library panel. You can edit the video symbol's properties in the Video Properties dialog box. For more information, see "Viewing and editing device video symbol properties" on page 76.

To import a device video by using the Library panel:

1. In Flash, open the Library panel (Window > Library).
2. Open the Library options menu and choose New Video.

   The Video Properties dialog box appears.
3. In the Video Properties dialog box, select the option to bundle the source video in the SWF file, then click Import.
4. Browse to the folder that contains the device video file and select it.

   If you don't see the desired video file listed in the Open dialog box (or if you can see it but can't select it), select All Files (*.*) from the Files of Type pop-up menu (Windows), or All Files from the Enable pop-up menu (Macintosh). This is necessary sometimes because the Flash authoring tool doesn't recognize most device video formats.
5. Click Open.

   In the Video Properties dialog box, select the Export for ActionScript option and enter a string in the Identifier text box to play the device video clip by using a separate video object on the Stage. For more information, see "Playing a bundled video directly from the library" on page 73.
6. Click OK to close the Video Properties dialog box.

   A new video symbol appears in your document's Library panel that's associated with the device video file. You can edit the video symbol's properties in the Video Properties dialog box. For more information, see "Viewing and editing device video symbol properties" on page 76.

Bundled device video example

This section shows you how to create an application that plays a bundled device video file. To do this, you'll first import a device video into your Flash document (see "Importing device video" on page 69), create an instance of the video on the Stage, and finally add ActionScript to play and stop the video.

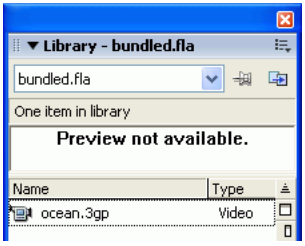
You can also play video directly from the library. For more information, see "Playing a bundled video directly from the library" on page 73.

To import and play a bundled device video:

1. In Flash, create a new document from the Flash Lite 2.0 Symbian Series 60 template, and save it as bundled_video.fla.
2. Import the device video file named ocean.3gp in the Samples and Tutorials folder located in the Flash CS3 installation folder on your hard disk.

   For more information about importing a device video, see "Importing device video" on page 69.

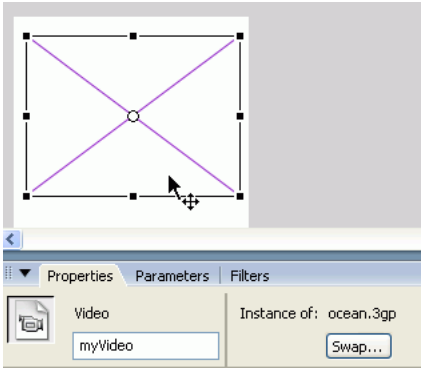
   A new video symbol appears in the Library panel.



3. In the Timeline, select the layer named Content, then drag an instance of the video symbol in the Library panel to the Stage.

4. Select the video object on the Stage and, in the Property inspector, type **myVideo** in the Instance Name text box.

   This is the name you will use to refer to the video object in ActionScript code.



5. In the Property inspector, set the video object's width to 176 and its height to 144.

   These dimensions match those of the source video. Depending on the device, a device video in Flash Lite does not always scale to fit the size of the video object's bounding box. For more information, see "Limitations of device video" on page 77.

6. To add buttons to control the video, open the library of prebuilt buttons (Window > Common Libraries > Buttons).
7. In the Buttons library, double-click the Circle Buttons folder to open it.
8. Drag an instance of the Play button symbol from the Buttons library to the Stage.
9. Drag an instance of the Stop button symbol from the Buttons library to the Stage.
10. Select the Play button on the Stage and open the Actions panel (Window > Actions).
11. Type (or copy and paste) the following code into the Actions panel:

```
on(press) {
  myVideo.play();
}
```

12. Select the Stop button on the Stage and type the following code in the Actions panel:

```
on(press) {
  myVideo.stop();
}
```

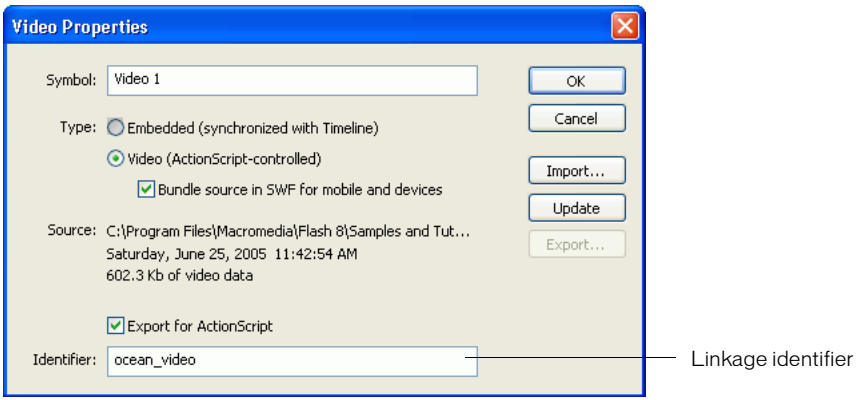
13. Publish the SWF file (File > Publish) and transfer it to your device for testing.

NOTE	You must test a device video on the actual device; you can't preview it in the emulator.

Playing a bundled video directly from the library

Earlier in this document (see "Using bundled device video" on page 69) you learned how to import and play a single bundled device video. To do this, you imported the device video into the library, added an instance of the video symbol to the Stage, and called the `Video.play()` method on the video instance.

You can also use a single Video object on the Stage to play multiple bundled device videos directly from the library. To do this, you bundle the device video in your application's library. You also assign an identifier to the video symbol that lets you reference the video symbol with ActionScript, as the following image shows:



You then create another placeholder video symbol and add an instance of it to the Stage. To use the placeholder video to play the device video in the library, you pass the symbol's ActionScript identifier to the `Video.play()` method, as the following example shows:

```
placeHolderVideo.play("symbol://ocean_video");
```

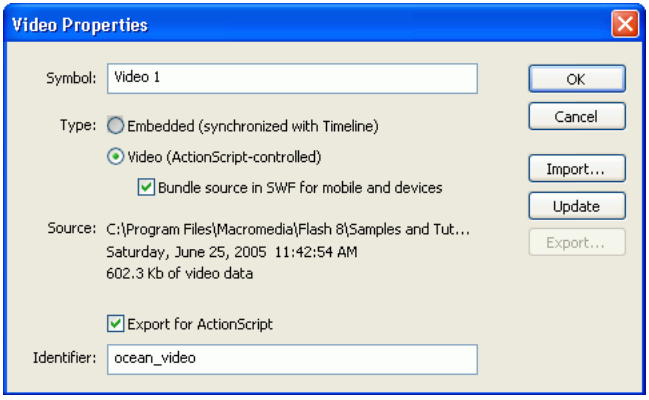
The following procedure demonstrates how to use this technique to play a single video directly from the library.

To play a video directly from the library:

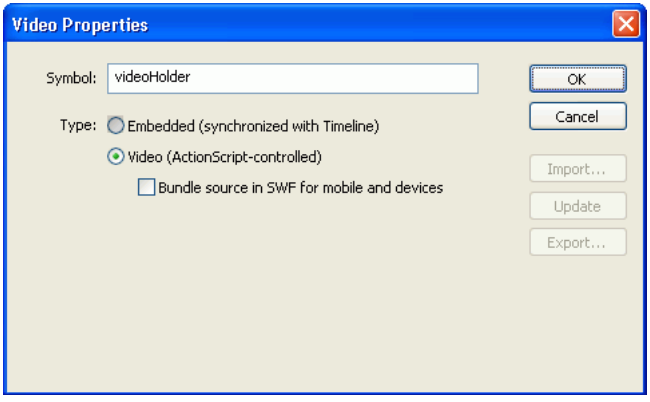
1. In Flash, create a new document from the Flash Lite 2.0 Symbian Series 60 template, and save it as **library_video.fla**.
2. Import the video file named ocean.3gp located at www.adobe.com/go/learn_flt_samples_and_tutorials. On the Samples and Tutorials page, locate, download and decompress the .zip file for your Flash Lite version, and then navigate to the Samples folder to access the file.

   For more information about importing device video, see "Importing device video" on page 69.

3. Right-click (Windows) or Control-click (Macintosh) the ocean.3gp video symbol in the library and choose Properties from the context menu. The Video Properties dialog box appears.
4. In the Video Properties dialog box, select Export for ActionScript and type **ocean_video** in the Identifier text box, as the following image shows:



5. Click OK to close the Video Properties dialog box.
6. To create the placeholder video clip, do the following:
   a. In the Library panel, click the options menu button in the panel's title bar and select New Video. The Video Properties dialog box appears.
   b. In the Symbol text box, type **videoHolder**.



   c. Click OK to close the Video Properties dialog box.
7. In the Timeline, select the layer named Content, then drag an instance of the videoHolder symbol to the Stage.

8. In the Property inspector, type `myVideo` in the Instance Name text box.
9. To add buttons to control the video, open the library of prebuilt buttons (Window > Common Libraries > Buttons).
10. In the Buttons library, double-click the Circle Buttons folder to open it.
11. Drag an instance of the Play button symbol from the Buttons library to the Stage.
12. Drag an instance of the Stop button symbol from the Buttons library to the Stage.
13. Select the Play button on the Stage and open the Actions panel (Window > Actions).
14. Type (or copy and paste) the following code into the Actions panel:

```
on(press) {
  myVideo.play("symbol://ocean_video");
}
```

15. Select the Stop button on the Stage and type the following code in the Actions panel:

```
on(press) {
  myVideo.stop();
}
```

16. Publish the SWF file (File > Publish) and transfer it to your device for testing.

NOTE	You should test device video on the target device; not all formats are supported by the emulator.

Using external device video

In addition to playing video that's bundled in the SWF file (see "Using bundled device video" on page 69), you can also play external video files that reside on the device's memory card or that come from a network address. As with bundled device video, the device is responsible for decoding the device video.

To play an external video file, you pass the absolute or relative file location or URL of the video file to the `Video.play()` method. In the following example, the SWF file and the 3GP file are located in the same folder on the device.

```
myVideo.play("ocean.3gp");
```

You can also specify a relative folder path to the SWF file, as follows:

```
myVideo.play("folder1/folder2/ocean.3gp"); //
```

Depending on the device, you can also use the `file://` protocol to play a video file at a specific location, as follows:

```
myVideo.play("file://c:/folder1/folder2/ocean.3gp");
```

NOTE

Not all devices support the `file://` protocol. Be sure to test your application on all target devices if you use this protocol.

Depending on the device, you can also load a video file from a network address. Current implementations support video streaming using the RTSP (Real Time Streaming Protocol) only, as shown in the following example. HTTP-based streaming is not supported.

```
myVideo.play("rtsp://www.adobe.com/video/ocean.3gp");
```

Viewing and editing device video symbol properties

The Video Properties dialog box lets you view and edit information for video symbols in the Library panel.

To open the Video Properties dialog box, do one of the following:

- Right-click (Windows) or Control-click (Macintosh) a video symbol in the Library panel and choose Properties from the context menu.
- Select a video symbol in the Library panel and choose Properties from the options menu in the panel's title bar.

To import a device video into a video symbol:

1. Select a video symbol in the Library panel and open the Video Properties dialog box.
2. In the Video Properties dialog box, select the option to bundle the video source in the SWF file, if it's not already selected.
3. Click Import and, in the file browser, locate the device video file to import and select it.

   If you don't see the desired video file listed in the Open dialog box (or if you can see it but can't select it), select All Files (*.*) from the Files of Type pop-up menu (Windows), or All Files from the Enable pop-up menu (Macintosh). This is necessary sometimes because the Flash authoring tool doesn't recognize most device video formats.
4. Click Open to close the file browser.
5. Click OK to close the Video Properties dialog box.

NOTE

The OK button is dimmed until you either import a device video or deselect the option to bundle the video source in the SWF file.

To assign an identifier to a video symbol:

1. Select the video symbol and open the Video Properties dialog box.
2. Import a device video into the symbol, if you haven't already.
3. Select Export for ActionScript.
4. In the Identifier text box, enter an identifier for the video symbol.

   Like all identifiers in ActionScript, the first character must be a letter, underscore (_), or dollar sign ($). Each subsequent character can be a number, letter, underscore, or dollar sign.
5. Click OK to close the Video Properties dialog box.

Determining supported video formats

Different devices support different video formats and encodings. Before developing a Flash Lite application that uses video, check the device manufacturers' specifications to determine what video formats are supported by the target devices.

You can also use the `System.capabilities.videoMIMETypes` property to determine what video formats a device supports. This property contains an array of video MIME types that the device supports. Each item in the array has the following format:

`video/`*`video-type`*

For example, the following code displays in a TextField object named `mimeTypes_txt` all of the video MIME types that the device supports:

```
var mimeTypes = System.capabilities.videoMIMETypes;
mimeTypes_txt.text = mimeTypes.toString();
```

The indices for the items contained in the `System.capabilities.videoMIMEtypes` array are equal to the supported device video MIME types. This provides a quick way to test whether a device supports a particular video format. For example, the following code checks whether the device supports 3GPP video before playing a video file of that type:

```
if (System.capabilities.videoMIMETypes["video/3gpp"]) {
  my_video.play("movie.3gp");
}
```

Limitations of device video

The following limitations apply to using device video in your Flash Lite 2.0 applications:

- You can't rotate or skew device video. Some devices may support scaling.
- You can't synchronize device video with the timeline.

- You can't composite or blend device video with other media. The device renders video directly to the display atop any other Flash content.
- You can't control the sound volume of a video clip.

Controlling video with ActionScript

To control device video playback, you use the ActionScript Video object. There are several differences between the Video object in Flash Lite 2.0 and the Video object in the desktop version of Flash Player.

The following methods of the Video object are available only in Flash Lite 2.0 and not in the desktop version of Flash Player:

- `Video.play()`
- `Video.stop()`
- `Video.pause()`
- `Video.resume()`
- `Video.close()`

> **NOTE** These methods of the Video object correspond roughly to the same methods available in the NetStream object in the desktop version of Flash Player.

The following methods and properties of the Video object in the desktop version of Flash Player are not supported by Flash Lite 2.0:

- `Video.attachVideo()`
- `Video.clear()`
- `Video.deblocking`
- `Video.height`
- `Video.smoothing`
- `Video.width`
- `Video._visible`

For more information about using the Video object in Flash Lite, see the `Video` object entry in *Flash Lite 2.x ActionScript Language Reference*.

Loading external images

In Flash Lite 2.0, as in Flash Lite 1.1, you can use the `loadMovie()` function (or, equivalently, the `loadMovie()` method of the MovieClip object) to load external SWF files into your application. Additionally, in Flash Lite 2.0, you can use the `loadMovie()` function to load any arbitrary image format that the device supports.

For example, assuming that the target device can decode PNG files, the following code loads and displays an external PNG file that resides on the web server:

```
loadMovie("http://www.adobe.com/image.png", "image_target");
```

To determine what image formats the target device supports, you can use the `System.capabilities.imageMIMETypes` property, which contains an array of supported image MIME types. The index of each element in the array is equal to each supported MIME type.

For example, the following ActionScript determines whether a device supports PNG images before the device attempts to load an external PNG file:

```
if (System.capabilities.imageMIMETypes["image/png"]) {
  loadMovie("images/image.png", "mc_myPngImage");
}
```

Flash Lite limits to five the number of `loadMovie()` operations that an application can perform in a given frame, and 10 total operations at any one time. For example, suppose your application contains code on frame 1 to load six external JPEG images, as shown below:

```
image1.loadMovie("image1.jpg");
image2.loadMovie("image2.jpg");
image3.loadMovie("image3.jpg");
image4.loadMovie("image4.jpg");
image5.loadMovie("image5.jpg");
image6.loadMovie("image6.jpg"); // Won't load
```

In this case, only the first five images (image1.jpg through image5.jpg) will load; the last image (image6.jpg) will not load because the five connection limit is reached. One solution is to split the `loadMovie()` calls over multiple frames so that each frame contains a maximum of five `loadMovie()` calls.

CHAPTER 5

Developing Flash Lite Applications for BREW

5

This chapter describes how to develop Adobe Flash Lite 2.x applications that can run on devices that use the Binary Runtime Environment for Wireless® (BREW®) platform made by QUALCOMM Incorporated. If you are a developer who is comfortable working with Flash Lite but are not familiar enough with the BREW platform and its requirements to readily produce applications for BREW devices, you should read this chapter. You will also find this chapter useful if you have expertise with other application development technologies and want to explore using Flash Lite to develop BREW applications.

This chapter contains the following topics:

BREW basics

The QUALCOMM BREW mobile device platform is installed and supported on a variety of devices made by different manufacturers worldwide. Adobe Flash CS3 Professional includes tools and resources that developers use to create Flash Lite-based applications and content that users can download and use on BREW devices.

Flash developers use the Flash authoring tool to create Flash Lite content and applications for BREW-enabled devices. Developers then submit their applications to National Software Testing Laboratories (NSTL) for True BREW Testing (TBT). Once applications pass TBT, they are uploaded to the BREW Delivery System (BDS), and if chosen for distribution by a carrier, they are made available on the carrier's application download server (ADS). Device users can then download and purchase these applications, taking advantage of key features in the BREW platform for over-the-air distribution and billing through the BDS.

Developers who want to experiment with BREW without incurring the costs associated with full-scale BREW development can elect not to register as authenticated BREW developers and download only the free SDK components, which are sufficient to produce and test an application using the BREW Simulator, but not to upload and test it on a device, or to simulate specific device characteristics with customized device packs.

BREW is intended to operate as an "ecosystem" to help support the development and delivery of wireless device content. Its aim is to make it easier for developers to create, distribute, and derive profit from wireless applications. This ecosystem consists of the following main groups of participants:

Developers use tools available in the BREW SDK to develop content with C++ combined with APIs, or with a BREW-compatible extension. The BREW SDK is a free download, but you must register with QUALCOMM to get access to the Tools Suite and SDK Tools.

National Software Testing Laboratories (NSTL) tests applications that developers submit, and if they pass, the applications are eligible for inclusion in the BDS. If they do not pass, developers can correct and resubmit their applications.

QUALCOMM maintains the BREW Delivery System (BDS), which is a web-based system that delivers content listed in QUALCOMM's catalog, along with pricing and billing information. Pricing can involve several models, including operator auction.

Operators (carriers) use the application download server (ADS) to offer content they select at retail prices for purchase by customers through an over-the-air (OTA) delivery mechanism.

Device manufacturers use the BREW client, which exposes a common set of application programming interfaces (APIs) for standardized development of wireless applications. The client also includes an application manager that users can purchase and use to manage BREW applications.

Flash Lite and BREW development tools

To develop and test Flash Lite applications for the BREW environment, you need to acquire and install specific hardware and software tools.

Hardware

- **Windows computer**

  For information about system requirements for the BREW SDK and the BREW tools suite, see the "System Requirements" section in *Starting with BREW* (available from the BREW website). For information about system requirements for Adobe software, see www.adobe.com/go/sysreqs.

- **BREW-enabled devices with test bit set**

  For information about how to acquire, activate, and test-enable BREW devices, see the "Handset Acquisition and Readiness" section of *Getting Started with BREW* (available from the BREW website). Note that devices that support BREW 2.x must be sent to QUALCOMM to be test-enabled; BREW 3x devices are pre-enabled.

- **Data cables**

  Data cables are required to upload Flash applications to the BREW devices you plan to support. Most devices come bundled with data cables, but if you do not have a cable, see the "Acquire BREW Handsets and Data Cables" section of *Getting Started with BREW* (available from the BREW website).

Software

- **BREW SDK and BREW Tools Suite**

  For information about installing the BREW SDK and tools, see Chapter 2, "Installing the BREW SDK," in *Starting with BREW* (available from the BREW website). You must be a registered BREW developer to download and install the BREW Tools Suite, but not the SDK itself. For instructions on how to register, see the BREW website.

- **USB drivers for targeted devices**

  Obtain and install the USB drivers for the devices you intend to target. For information about how to acquire the drivers, contact the device manufacturer.

- **Flash Lite extension for BREW-enabled devices** (available from the BREW website for testing purposes only.)

  The extension is downloaded automatically to users' devices when they select and download a Flash Lite for BREW application.

  These components are described in detail in ""Setting up your system for BREW" on page 87.

Flash Lite features that BREW supports

Flash Lite for BREW supports a subset of the features available in both Flash Lite 2.0 and Flash Lite 2.1. The features that Flash Lite for BREW supports are as follows. For further information about these features, see "About Flash Lite 2.x features" on page 7.

- Built on Macromedia® Flash® Player 7 from Adobe, supports ActionScript 2.0
- XML data handling
- Persistent data management
- Device, vector fonts
- Device sound for events
- Inline video (restricted to the video formats that are supported on the device)
- Streaming video
- XML sockets
- Network access and HTTP streaming
- Inline text support (limited to 64,000 characters, with further limitations imposed by individual device implementations. Some devices support as few as 1000 characters.)
- Dynamic multimedia
- Keypad and keyboard support

Flash Lite features that BREW does not support

Manufacturers of BREW devices can limit the use of some Flash Lite features, which means that these features cannot be used on the device. Information about specific devices is available in the BREW website's Developer Resources area.

In addition to features that are not supported on a given device, the following list describes the functionality that is not supported in the current Flash Lite BREW implementation (and is thus not available on any device):

Flash Lite sound decoding All sound formats are passed to the device for playback without modification. For example, if the sound is in MP3 format and the device supports MP3, it plays. Flash Lite does not decode any sounds internally in BREW, as it does on other platforms.

Streaming sound All sound must be fully loaded before it can play. The current BREW implementation does not support playing sounds progressively as they are loading over a network.

ActionScript commands The following ActionScript `fscommand2()` functions are unavailable on any BREW device:

`ExtendBacklightDuration()`	`GetNetworkConnectStatus()`
`GetBatteryLevel()`	`GetNetworkName()`
`GetMaxBatteryLevel()`	`GetNetworkRequestStatus`
`GetMaxSignalLevel()`	`GetPowerSource()`
`GetNetworkConnectionName()`	`GetSignalLevel()`

The following ActionScript `fscommand()` function is unavailable on any BREW device:

`fscommand(Launch)`	

getURL() support Some BREW handsets support some `getURL()` functions, depending on the implementation. As a general rule, the following protocols are not supported:

- HTTP (no API exists to enable Flash Lite to launch a browser)
- Multimedia Message Service (MMS)

NOTE

`mailto` requests are not directly supported. Instead, you can use the Short Message Service (SMS) protocol, which limits the maximum message size to 160 characters.

Wallpaper content Flash Lite wallpaper content is not supported in the Flash Lite for BREW implementation.

Animated ring tones Animated ring tones are not supported in the Flash Lite for BREW implementation.

BREW SDK versions supported

Flash Lite 2.1 for BREW Devices currently supports BREW versions 2.1.3 and later. You can find the version of BREW that is running on a particular device in the Device Specifications list on the BREW website. The two devices that currently support Flash Lite for BREW (the Samsung SCH-A950 and the LG VX9800) support different versions of BREW (2.x and 3.x, respectively). Because the BREW SDKs are backward-compatible, you can download the SDK and tools for the most recent version (3.x at this writing) rather than the version supported on the device you are targeting for development. For example, if you are targeting the Samsung SCH-A950, which is a 2.x device, you can still download and use the 3.x version of the BREW SDK and Tools Suite.

The differences between BREW 2.x and 3.x are minimal in terms of the user interface, and Flash Lite behaves the same on both platforms. The significant differences are in the way files are stored on the device and in how applications in the BREW Tools Suite are loaded. For more information about file system structure, see "Device file structures for different BREW versions" on page 97.

Devices supported

The devices that currently support the BREW platform are:

Device	Version supported
Samsung SCH-A950	BREW version 2.1.3
LG VX9800	BREW version 3.1.2

The Adobe customer support team will certify additional devices as they become available. For a current list of supported devices, see www.adobe.com/go/mobile/supported_devices/.

Where to find more information

For information about BREW, see the BREW website. The Developer Home area of the site contains information about the BREW SDK, BREW tools and utilities, and the characteristics of specific devices that support BREW.

NOTE

You must be a registered BREW developer to download and use the BREW Tools Suite, as well as to enable your devices for testing BREW. Details about how to register are available on the site.

Setting up your system for BREW

This section describes how to set up your system to develop Flash Lite applications that can run on devices that use the BREW platform. To set up your system to produce Flash Lite files for BREW, you need to locate and install the appropriate software from several sources. You also need to acquire the appropriate hardware (BREW-enabled devices, data cables, and so on). Much of the specific information required to install and configure these components is located in other Flash Lite or BREW documents. Therefore, rather than repeating information available elsewhere, this section provides an overview of the installation and configuration steps combined with references to the appropriate sources for detailed information on how to complete each setup task.

This section contains the following topics:

Workflow for setting up your system for BREW

To set up your system to support Flash Lite authoring for BREW, you complete the following tasks:

1. Register as a BREW developer.
2. Install the BREW SDK and Tools Suite.
3. Install Flash Lite 2.1 for BREW.
4. Install USB drivers for target devices.

Registering as a BREW developer

To register as a BREW developer, go to the QUALCOMM's main BREW site and follow the links on the Developer home page to become an authenticated BREW developer.

Installing the BREW SDK and tools

The instructions that follow describe how to install the BREW SDK on your computer. They assume that you have already registered as a BREW developer. For more detailed information about installing the BREW SDK or Tools Suite, see "Installing the BREW SDK" in *Starting BREW*. Go to the BREW website and then choose Developer Home > BREW Documentation > Application Development Documentation.

To install the BREW SDK and Tools Suite:

1. From BREW Developer Homeon the BREW website, click Download BREW Tools to display the BREW Development and Commercialization Tools page.
2. Navigate to the BREW Development area and click the appropriate link to download the most recent BREW SDK version (at this writing, BREW SDK 3.1).

   Follow the onscreen instructions to download and install the SDK.
3. After you install the SDK, install the tools.

   Navigate to the BREW Commercialization area of the BREW Development and Commercialization Tools page and click the link for the BREW Testing and Commercialization Utilities. Follow the onscreen instructions to install the Tools Suite on your computer.

   Your Start menu should now contain entries for both the BREW SDK and the BREW Tools Suite.

Installing Adobe Flash Lite 2.1 for BREW Devices

Flash Lite 2.1 for BREW Devices includes three components:

- Flash Lite Publisher for BREW (available from http://www.adobe.com/go/support_flashlite)
- Flash Lite 2.1 for BREW Simulator (available from http://www.adobe.com/go/support_flashlite)
- Flash Lite extension for BREW-enabled devices (available from the BREW website for testing purposes only. The extension is downloaded automatically to users' devices when they select and download a Flash Lite for BREW application.)

Installing USB drivers for devices

Each device you intend to target for BREW development includes its own set of USB drivers, which must be installed on your computer. New devices should include a CD-ROM that contains these drivers; use these versions whenever possible. To install USB drivers for a device, follow the standard procedure for installing device drivers in Windows.

Where to find more information

For more information about Flash Lite application development and testing, see the other sections in this document, which provide comprehensive information about how to develop and test Flash Lite applications. The online Help for Flash CS3 Professional also contains extensive information on Flash Lite development.

For more information about BREW, the best source of information is the QUALCOMM BREW website.

You can also consult the Help files included in the SDK for detailed information about BREW development processes and tools.

At this writing, the BREW documentation set includes two similarly named documents, both of which are useful for learning BREW basics. These documents are:

- *Getting Started with BREW* (a two-page summary of the hardware and software you need to develop BREW applications).
- *Starting with BREW* (a more comprehensive document that contains all of the information you need to start developing BREW applications).

Authoring Flash Lite files for BREW

Authoring for BREW is similar to authoring for Flash Lite in general, except that you tailor your applications to the characteristics of the specific BREW-enabled handset for which you are designing. This section describes how to use Flash to author Flash Lite content that can later be published for BREW by using the Flash Lite Publisher for BREW (described in "Publishing Flash Lite files for BREW" on page 93).

This section contains the following topics:

Workflow for authoring Flash Lite files for BREW

Creating Flash Lite content for the BREW platform is an iterative process that involves the following tasks:

Identify your target devices and Flash Lite content type At this writing the following two devices support BREW: the Samsung SCH-A950, which includes the BREW version 2.1.3 platform; and the LG VX9800, which includes the BREW version 3.1.2 platform. These devices have substantially different characteristics (for example, the LG has a QWERTY keyboard and the Samsung does not), so it is important to tailor your application development plans to the device's unique capabilities.

Also, you can create two different content types—applications and screen savers; each content type requires different application design concerns. For more information about how screen savers differ from applications, go to the BREW website and navigate to Developer FAQs -> BREW Tools -> MIF Settings -> Screensaver.

NOTE	Currently only the Samsung SCH-A950 supports screen saver content types.

Create and test your application in Flash Adobe Flash CS3 Professional includes an emulator on Adobe Device Central CS3 that lets you test your application without having to transfer it to a device. You use the Device Central emulator to refine your application's overall design and fix any problems before you test it on a mobile device.

NOTE	The testing you do in Flash using the emulator is different from the testing you do using the Simulator that is included with the BREW SDK. Flash emulator testing is described in this section, and BREW Simulator testing is described in "Publishing Flash Lite files for BREW" on page 93. Because the Device Central emulator does not currently provide device packs for BREW devices, the Device Central emulator can be used only to test basic functionality. Use the BREW Simulator to test your application's functionality in the BREW environment.

Identifying your target device and content type

The QUALCOMM Developer Home contains information about the specific characteristics of all of the devices that currently support a version of BREW. Some information is also available about precommercial devices that are not yet available on the market. You can use this information to determine the devices that you plan to target with your BREW application. Several sample applications with different device packs are also included with the BREW SDK. You can use these sample applications to learn more about how specific devices work.

Creating your application in Flash

The process for creating Flash Lite applications for BREW is similar to creating a generic Flash Lite application. With the exception of features that are not supported on BREW devices (listed in "Flash Lite features that BREW does not support" on page 84) and BREW-specific authoring notes (below), developing applications for BREW devices can be done by following the development steps detailed in the rest of the Flash documentation.

The following topics provide information that applies only to Flash Lite authoring for BREW.

Launching a browser

The BREW target handsets use the OpenWave browser, which doesn't provide the appropriate hook for the BREW application to invoke the browser. Therefore, if you specify an HTTP URL in the `getURL()`, `loadVars(send)`, or `XML.send()` functions, no browser is launched. To send information to a server and continue playing your SWF file without opening a new window or replacing content in a window or frame, use the `loadVars.sendAndLoad()` function.

Default soft key behavior

For information and examples describing how to program soft key behavior in Flash Lite, see "Using the soft keys" on page 32. The following table describes the default soft key behavior in Flash Lite 2.1 for BREW:

Content action	Screen mode	Left key action	Right key action
Default (content does not call the `SetSoftKeys` command)	Full-screen mode	Displays options menu	Exits the player
Non-full-screen mode (content does not call `SetSoftKeys`)	Non-full-screen mode	Displays options menu	Exits the player
Content disables soft keys	n/a	None	None
Content overrides one or both soft keys (calls `SetSoftKeys`)	n/a	Displays options menu if no override. If override, behavior specified by content.	Exits the player if no override. If override, behavior specified by content.

Testing your application in Flash

As with creating your application, using the Adobe Device Central emulator to test your BREW application is similar to testing a generic Flash Lite application. Remember that the testing you do in Flash (using the Device Central emulator) is different from the testing you do using the Simulator that is included with the BREW SDK. Device Central emulator testing is described in this section, and BREW Simulator testing is described in "Publishing Flash Lite files for BREW" on page 93

Because the Device Central emulator does not currently provide device packs for BREW devices, the Device Central emulator can be used only to test basic functionality. Use the BREW Simulator to test your application's functionality in the BREW environment.

To use the Adobe Device Central emulator to test your application:

1. Open your application in Flash.
2. Select File > Publish Settings. On the Flash tab, select Flash Lite 2.1 from the Version list, and then select ActionScript 2.0 from the ActionScript version list. Click OK to save your settings.
3. Launch the Adobe Device Central emulator (Control > Test Movie) and click the Device Profiles tab.
4. In the Available Devices pane, do one of the following:
   a. If you have not downloaded a device pack for your target device, select Generic > Flash Lite 2.0 > Generic Phone.
   b. If you downloaded a device pack for your targeted device (available from www.adobe.com/products/flash/download/device_profiles), click the device name to expand it, select the appropriate device size, and then drag the name of your targeted device from the Available Devices pane to the Device Sets pane.
5. Select File > Return to Flash, and then select Control > Test Movie on the Flash menu. If the BREW Publisher wizard appears, click Cancel to close the dialog box. Flash exports the application to Device Central and opens it in the emulator.
6. Use the emulator to test your application.

   For information about how to use the emulator, see "Using the emulator" on page 116.

   When you are finished testing your application in the emulator, save your files and proceed to "Publishing Flash Lite files for BREW" on page 93 for instructions on how to publish your application for the BREW platform.

Where to find more information

For further information about developing and testing Flash Lite applications, consult the following sources, which you can find in Flash Help:

- *Getting Started with Flash Lite 2.x* (PDF)
- Testing Flash Lite Content (Help topics)

Publishing Flash Lite files for BREW

This section describes how to use the Flash Lite Publisher for BREW wizard to generate Flash Lite content that can run on devices that use the BREW platform. It assumes that you already understand how to author in Flash Lite, and covers only the publishing of previously created Flash Lite files so that they can be used on BREW-enabled devices.

This section includes the following topics:

About publishing Flash Lite files for BREW

To run on BREW devices, Flash Lite applications must conform to the BREW standard. The Flash Lite for BREW Publisher wizard makes it easier for developers to create applications that conform to this standard by automating some of the steps required. In addition to generating the files for the application itself, the wizard also uses the Simulator tool in the BREW SDK to generate files that you can use to test your BREW application.

Before you start using the BREW Publisher wizard, it is useful to understand some basic information about BREW application file types, file structure, and requirements.

File types

The primary purpose of the Flash Lite Publisher for BREW is to publish a standard SWF file, and then use a post-processor to package the SWF file so that it can run on a BREW device. The following table lists the files needed to use Flash Lite content on a BREW device:

File	File content
MIF file	Configuration information
MOD file	Application file for BREW device
SIG file	Device-specific file from QUALCOMM

The files needed to use Flash Lite content with the BREW Simulator are as follows:

File	File content
MIF file	Configuration information
DLL file	Application file for BREW Simulator

All of these files must have the same filename (plus the appropriate file extension: .mif, .mod, .sig, or .dll), and the filename must also be the same as the application folder name. These files must also be located in the correct folders on the device in order to work. The folder structure varies according to the version of BREW your target device uses. For more information, see "Device file structures for different BREW versions" on page 97.

MIF files

MIF files are configuration files that provide information required by BREW applets. The BREW Publisher wizard generates MIF files for you as part of the publishing process. However, it is the responsibility of the individual content developer to understand the MIF format and requirements, and to verify that the MIF file settings are correct before submitting applications for True BREW Testing.

The following code shows the default MIF values that are output by the BREW Publisher wizard, for applications and screen savers, respectively. If any of these values are incorrect for your application, download the MIF Editor tool from the BREW website and edit the files before submitting your applications for testing.

```
<Applet>
- List of Applets defined in this Module: [Class ID in post processor]
- Applet Information:
* Name: [Applet Name in post processor]
* Class ID: [Class ID in post processor]
* Applet Type: [blank]
```

```
- Graphics
* Large: [Large Icon in post processor]
* Medium: [Medium Icon in post processor]
* Small: [Small Icon in post processor]
<Extension>
- Exported Classes: [blank]
- Exported MIME Types: [blank]
<Dependencies>
- External Classes Used by this Module: [blank]
- External Classes Directory (Checkbox): [checked]
- Used: [Class IDs generated by post processor]
<Privileges>
File [checked]
Network [checked]
Web Access [checked]
TAPI [checked]
<Access Control>
- ACL type [blank]
- Rights [blank]
- Groups [blank]
<License>
No License (Checkbox): [checked]
<General>
- Author Name: [Author in post processor]
- Copyright String: [Copyright in post processor]
- Module Version: [Version in post processor]
- EFS Restrictions
* Max Files: [blank]
* Max Space: (bytes) [blank]
- String Encoding Format: [ISOLATIN1]
Default MIF output for Screensavers
<Applet>
- List of Applets defined in this Module: [Class ID in post processor]
- Applet Information:
  * Name: [Applet Name in post processor]
  * Class ID: [Class ID in post processor]
  * Applet Type: [blank]
- Graphics
  * Large: [Large Icon in post processor]
  * Medium: [Medium Icon in post processor]
  * Small: [Small Icon in post processor]
- Notifications, Flags, Settings…
  * Flags: Screensaver [checked]
<Extension>
- Exported Classes: [blank]
- Exported MIME Types: [brew/ssaver Class ID]
<Dependencies>
- External Classes Used by this Module: [blank]
- External Classes Directory (Checkbox): [checked]
- Used: [Class IDs generated by post processor]
```

```
<Privileges>
All unchecked
<Access Control>
- ACL type [blank]
- Rights [blank]
- Groups [blank]
<License>
No License (Checkbox): [checked]
<General>
- Author Name: [Author in post processor]
- Copyright String: [Copyright in post processor]
- Module Version: [Version in post processor]
- EFS Restrictions
  * Max Files: [blank]
  * Max Space: (bytes) [blank]
- String Encoding Format: [ISOLATIN1]
```

SIG files

SIG files are device-specific signature files that are required for each device you are using for your application development. These files must be obtained from QUALCOMM and are good for 90 days. After 90 days, the signature file expires and QUALCOMM must issue a new signature file.

Only one SIG file is required per device. However, because each individual application requires its own corresponding SIG file, you make a copy of the original SIG file and rename it to match the name of the application you have created. Remember that the SIG file must reside in the same folder as the MOD file, the application file for the BREW device.

You must be an authenticated BREW developer to access the signature generation utility on the BREW website.

NOTE	SIG files are required for testing applications on devices, but they are not required for testing using the BREW Simulator.

To obtain a SIG file:

1. Go to the BREW website and navigate to the page that contains the link to the QUALCOMM web-based test signature generator.
2. Select BREW Testing Generator.
3. Enter your BREW developer user name and password.
4. In the ESN field, type **0x<ESN Number>** where the ESN number is the number you find under the device battery when you lift it up.
5. Click Generate and wait for the Download Signature link to appear.
6. Download the signature and save the file; it will be required for all testing on the device.

BAR files

You can add, delete, and move files on a BREW-enabled device with a data cable and a Windows application called AppLoader (QUALCOMM provides AppLoader as part of the BREW Tools Suite). However, you cannot copy files from the BREW device that have the extensions .mod, .mif, or .bar to any other device or storage medium. This limitation is enforced by QUALCOMM to ensure that files purchased through the BREW Delivery System (BDS) cannot be distributed to unauthorized users.

To enable Flash content developers to take advantage of this security feature, the Flash Lite player that runs on BREW devices checks for BAR files when it attempts to reference a file on the local device. This restriction applies to any ActionScript or Shared Libraries function that uses a path to reference a file (for instance, `loadMovie()`, `loadSound()`, `loadVariables()`, `XML.load()`, and so on). To use this feature, developers can append the .bar extension to any files they create, and the player can still locate and display them, but no one can copy them from the BREW device onto which they were initially loaded. The Flash Lite player first checks for the existence of the file by using the supplied local path (for instance, loadMovie(*example*.swf), and if the file is not found at this location, the player than checks the same path with the .bar extension appended (for instance, loadMovie(*example*.swf.bar)). The file is then loaded.

Device file structures for different BREW versions

Devices that support BREW 2.x and BREW 3.x have different file structures. It is important to understand these differences before you use the BREW Publisher wizard to specify your BREW application's file structure.

The file system on BREW 2.x devices is structured as follows:

- All application folders are stored in the root folder, called brew/.
- Application files (including SIG files) are stored in the application folder (subfolders are not allowed).
- MIF files are stored in the root folder, brew/.

- Shared media, such as images and BAR files are stored in the brew/shared folder. You can also store shared media in the folder that contains the main application files.

  Example:

  brew/*appname*/*appname*.mod

  brew/*appname*/*appname*.sig

  brew/*appname*.mif

  brew/shared/*media*.xxx, where xxx is the extension of any resource files, including graphics, video, sound, and BAR files

  or

  brew/*appname*/*media*.xxx, where xxx is the extension of any resource files, including graphics, video, sound, and BAR files

The file system on BREW 3.x devices is structured as follows:

- All application directories are stored in the brew/mod folder.
- Application files (including SIG files) are stored in the application folder (subfolders are not allowed).
- MIF files are stored in the brew/mif folder.
- You can store shared media, such as images and BAR files, in the brew/shared folder. You can also store shared media in the folder that contains the main application files.

  Example:

  brew/**mod**/*appname*/*appname*.mod

  brew/**mod**/*appname*/*appname*.sig

  brew/**mif**/*appname*.mif

  brew/**shared**/*media*.xxx, where xxx is the extension of any resource files, including graphics, video, sound and BAR files

  or

  brew/**mod**/*appname*/*media*.xxx, where xxx is the extension of any resource files, including graphics, video, sound, and BAR files

Workflow for publishing and testing Flash Lite files for BREW

The workflow for publishing and testing Flash Lite files for BREW is as follows:

- In Flash, launch the BREW Publisher wizard and follow the screens to specify the appropriate settings for your BREW application and publish the files.
- In the BREW SDK, use the BREW Simulator to test your the application.

Using the BREW Publisher wizard to publish application files

The following steps assume that you already created and saved your application files in Flash. If you did not yet save your FLA file, the BREW Publisher Wizard prompts you to save it.

To use the BREW Publisher wizard to publish application files:

1. In Flash, select File > Publish Settings.

   The Publish Settings dialog box appears.
2. On the Formats tab, select Flash (.swf) if it is not already selected.

   You can deselect all other file types, because only the SWF file is needed to generate the BREW applet.
3. On the Flash tab, from the Version list, select a supported version of the player (Flash Player 7 or earlier, or any version of Flash Lite), and then select any of the versions from the ActionScript version list (all ActionScript versions are supported).
4. From the Post-Processor list near the bottom of the Flash tab, select Flash Lite for BREW.

NOTE	If the Post-Processor list does not appear on the Flash tab, check the instructions in "Setting up your system for BREW" on page 87 to ensure that you correctly installed Flash Lite 2.1 for BREW. The Post-Processor list appears only after the software is installed.

5. Click Settings to the right of the Post-Processor list to display the BREW Publisher wizard.
6. (Required) Use the input fields on the Identify Applet screen to specify a unique class ID and name for your BREW applet.

   The following table lists detailed information about these fields. When you are finished specifying your applet's class ID and name, click Next to continue.

Field name	Field contents
Class ID	This field is prepopulated with a randomly generated value as a default. Replace this value with your certified class ID if you have one. A class ID is a unique 32-bit identification code (8-digit hexadecimal value), which the BREW interface creation mechanism uses. For testing on a local system, you can use the randomly generated default (provided it is unique), although QUALCOMM must issue a unique ID for applications intended for distribution. For more information on obtaining class ID files, see the BREW website.
Applet Name	This field is prepopulated with the root portion of the name of the FLA file that is currently active. For example, if the active FLA file is called BREW.fla, the applet name is BREW. The name you enter in this field is displayed in the BREW Application Manager on the target device. Use the following guidelines for Applet names: • Applet names must be all lowercase, and must begin with an alphabetic character. • Numbers are allowed, but not as the first character. • The underscore character is the only special character allowed, and it may not be the first character of the name. • The maximum length of the name varies depending on the device. For information, see the device specifications on the BREW website.

7. (Optional) Use the Include Applet Information screen to enter information that you want to publish along with your BREW applet.

   The information that you enter here is stored in your applet's MIF file. The following table lists detailed information about each field. When you are finished specifying your information, click Next to continue.

Field name	Field contents
Author	Enter the applet author's name (23 characters maximum).
Version	Enter the applet version (23 characters maximum).
Copyright	Enter the copyright information (23 characters maximum).
Domain URL	Enter the fully qualified domain for your applet, ending with a trailing slash (example: http://www.example.com/). The applet's domain controls which information the applet has access to at run time. In earlier versions of the Flash player, SWF files from similar domains (for instance, www.exampleapps.com and www.examplegames.com) could communicate freely with each other and with other documents. In Flash Player 7 and later, the domain of the data to be accessed must match the data provider's domain *exactly* for the domains to communicate. For more information about domains and security, see the Flash documentation's topics on security, cross-domain security, and allowing data access between cross-domain SWF files.
Application Type (Application or screen saver)	Specifies whether the BREW applet you are creating is an application or a screen saver. For more information about how screen savers differ from applications, go to the BREW website and navigate to Developer FAQs › BREW Tools › MIF Settings › Screensaver.

8. (Optional for testing your application, but mandatory for applications that must pass True BREW Testing) Use the Applet Icons screen to enter or browse for the name and location of one or more icons for your applet.

   Supported file types are: JPEG, BMP, PNG, and BCI (BREW Compressed Image). These icons are displayed along with the applet's name in the BREW Application Manager on the target device. When you are finished specifying your icons, click Next to continue.

Field name	Field contents
Small icon	Image file for small icon (displayed on device). Maximum size: 16 x 16.
Medium icon	Image file for medium icon (displayed on device). Maximum size: 26 x 26.
Large icon	Image file for large icon (displayed on distribution server am some devices). Maximum size: 65 x 42.

9. Use the Define Output Settings screen to specify the locations in which to publish files for the BREW Simulator and the target device.

   These fields are populated with default values derived from your SWF file's location, combined with the applet's name and folder names (simulator and device) that indicate which files to use for testing and which files to upload to the device. You can accept the defaults, or you can specify different output locations. When you are finished, click Next to continue.

Field name	Field contents
For Simulator • MIF folder • Applet folder	Contains the files you need to test your applet by using the Simulator tool in the BREW SDK. For further information about the expected file locations and folder structure, see the documentation for the BREW Simulator on the BREW website.
For Device	Contains the files you upload to the BREW device. For more information about how folder structures differ according to the version of BREW supported by the target device, see "Uploading files to BREW devices" on page 105.

10. Use the Summary of Your Selections screen to review the applet settings you selected.

    To change any of these settings, use the Back button to return to previous screens and make changes. When your settings are complete, check or uncheck the box at the bottom of the screen to specify whether to display the wizard again when you publish, and then click Finish to save your settings.

Field name	Field contents
Summary of Your Selections	Displays a list of all of your settings to review.
Do not show this wizard on publish	Controls whether to display the wizard again when you publish the applet files, or trigger the publishing process without redisplaying the wizard. Note that you can always access the wizard regardless of this setting by displaying the Publish Settings dialog box and clicking the Settings button to the right of the Post-Processor list.

11. Click OK to save your settings, and then select File > Publish Settings again. The Publish Settings dialog box appears.

NOTE	This step is required only the first time you select Flash Lite for BREW from the Post-Processor list in a given file. Thereafter, you can publish without clicking OK first.

12. Click Publish to publish your files to the locations you specified using the wizard.

Using the BREW Simulator to test your application

Before uploading your application to the device, you should test it using the Simulator tool in the BREW SDK. For further information about this tool, see the documentation for the BREW Simulator on the BREW website.

To test your application with the BREW Simulator:

1. Navigate to the Simulator folder for the applet that you created by using the BREW Publisher wizard.
2. Open the Simulator folder and copy all of its contents to the BREW SDK examples folder.

> **NOTE** If you specified the BREW SDK examples folder as the location for the Simulator output files when you used the BREW Publisher wizard to publish the files, the files are already located in the correct folder and do not need to be copied.

3. Choose Start > Programs-BREW SDK v<*your version*> and select the Simulator for BREW to open the application.
4. In the Simulator, select File > Load Device, and then select the device pack for your target handset.
5. On the Simulator's Properties tab, select your application's folder as the Applet Directory.
6. Use the Simulator's tools and features to test your application.

Device packs

If you are an authenticated BREW developer, you can download device-specific device packs for the devices you are targeting from the Developer Resources area of the BREW website. The device pack for the Simulator may not exactly match your actual device. For example, the Samsung SCH-A950 device pack shows a BACK key, which is labeled as CLR on the actual device. Also, the device pack for the LG VX9800 does not support key input from the alphabetic keys on the keyboard, although the actual device does. Contact QUALCOMM for more information about specific device packs, updates, or issues.

If you are not an authenticated BREW developer, you can use the default device packs that are included with the SDK download. However, because these default device packs have less memory than the device-specific ones, you may need to select the device pack with the most memory (Device Pack 3 at this writing), or increase the memory to approximately 10,000,000 bytes to avoid performance issues and errors.

To increase the memory for a single Simulator session:

1. Open the Simulator and select the Device Details tab.
2. Scroll to the Memory section.
3. Increase the value for Heap Size (bytes). Note that this value is called RAM Size in some versions of the Simulator.

To make the memory increase persist across Simulator sessions:

1. Use a text editor to open the DSD file for the device pack you want to use (devicepack<*x*>.dsd, where *x* is the device pack number.)
2. Locate the appropriate setting and increase the value of Text:

```
<String Id="24" Name="IDS_DD_HW_RAM_SIZE">
     <Text>1048576</Text>
```

3. Save and close the file.

You may also encounter difficulties using the Simulator if the space allocated to the Examples folder and all of its subfolders is insufficient to allow your application to function properly. You can either delete unnecessary files from these folders manually, or increase the amount of space allocated for these files.

To make the Examples file size increase persist across Simulator sessions:

1. Use a text editor to open the configuration file and edit the appropriate value:
   - For the Samsung SCH-A950: The configuration file is SCH-A950.qsc. The value to be changed is shown in boldface:

     ```
     FS_LIMITS_PER_MODULE 65535 15204352
     ```

   - For the LG9800: The configuration file is LG-VX9800.dsd. The value to be changed is shown in boldface:

     ```
     <String Id="20" Name="IDS_DD_HW_EFS_SIZE">
     <Text>47000000</Text>
     <Comment>50MB</Comment>
     </String>
     ```

2. Save and close the file.

Where to find more information

For further information about the BREW Simulator, consult the following sources:

Document name and type	Location
Starting with BREW, "About the BREW Tools Suite" (PDF)	Go to the BREW website, and then select Developer Home › Application Developer Documentation.
BREW Simulator (Help)	Open the Simulator and click Help in the menu bar to display the Help.

Uploading files to BREW devices

This section describes how to upload Flash Lite applications to BREW devices, and how to test these applications on the handset.

This section contains the following topics:

About uploading Flash Lite applications for BREW

After you create your Flash Lite application and use the Flash Lite for BREW Publisher wizard to generate BREW-compatible files, you upload your application to a supported BREW device and test it.

Ideally you already targeted the device in your initial design process and tested your application by using the Adobe Device Central emulator and the BREW Simulator. However, it is still important to test your application thoroughly on the target device to ensure that it functions as you expect.

Prerequisites

Before you can upload your application, you need:

- A device that supports Flash Lite for BREW (at this writing, either a Samsung SCH-A950 or an LG VX9800)
- A serial or USB cable to upload your application from your computer to the device (usually available from the device manufacturer)
- The AppLoader tool (available for download as part of the BREW Tools Suite to authenticated developers from the BREW website
- Application files in the required file formats and structure (created by using the Flash Lite for BREW Publisher wizard)
  - ***appname*.SIG** (available for download to authenticated developers from the QUALCOMM web-based test signature generator
  - ***appname*.MOD** (generated by the Flash Lite Publisher for BREW wizard)
  - ***appname*.MIF** (generated by the Flash Lite Publisher for BREW wizard)
  - All sound, video, image, and BAR files that your application requires

  For complete information about required file formats and structure for your target device, see "Device file structures for different BREW versions" on page 97

Workflow for uploading applications

This workflow assumes that you have already installed the appropriate USB drivers for your device on your computer. For more information, see "Setting up your system for BREW" on page 87.

To upload and test Flash Lite applications on BREW devices, complete the following tasks:

- Connect handset to computer with cable.
- Open AppLoader on the computer:
  - Copy application files to the appropriate locations for the device. For information, see "Device file structures for different BREW versions" on page 97
  - If necessary, copy Flash Lite Extension and Player files to device (first time only)
  - Restart the device.
- Test the application on the device.

The rest of this section contains more detailed information about each of these tasks.

Uploading Flash Lite Extension files to the device (first use only)

Before you can test your Flash Lite for BREW applications on a given device, you copy the Flash Lite Extension onto the device. This needs to be done only once, when you first acquire the device for testing. End users of your application will receive these files as part of the download when they download your completed application from the BDS or ADS, but developers must copy these files manually. The files are listed below. Follow the instructions in the rest of this section to upload the files to your device using the AppLoader tool. Note that the locations to which you copy these files are different for BREW 2.x and 3.x devices. See "Device file structures for different BREW versions" on page 97 for more information.

Extension file	Source
flashlite_2_1.sig	Generated by developers using a tool available from the BREW website.
flashlite_2_1.mif	Available for download from the BREW website.
flashlite_2_1.mod	Available for download from the BREW website.
brewsaplayer.sig	Generated by developers using a tool available from the BREW website.
brewsaplayer.mif	Available for download from the BREW website.
brewsaplayer.mod	Available for download from the BREW website.

AppLoader hints and best practices

The following is a list of hints for using the AppLoader tool:

- When you create a folder on the device, do not use the backward slash (\).
- Do not run any applications on the device while you are copying files.
- Restart the device after you use AppLoader to make any changes.
- Avoid overwriting files in AppLoader; instead, delete the old versions of files on the device and replace them with new copies of the files.
- Do not upload files or folders whose names consist of numerals only. You will be unable to delete these files and folders from some devices.
- Do not begin a file or folder name with the characters "shared." "Shared" is a reserved word in BREW, and you will be prevented from uploading files or folders that begin with those characters.

Uploading applications to a BREW 2.x device

The following instructions explain how to upload a Flash Lite application for BREW to a Samsung SCH-A950, which includes the BREW version 2.x platform.

This procedure assumes that you have already installed the appropriate USB drivers for your device on your computer. For more information, see "Setting up your system for BREW" on page 87

This procedure also assumes that you have already uploaded the required Flash Lite Extension and Flash Lite player files to the device. This needs to be done only once, when you first begin using the device. For more information, see "Uploading Flash Lite Extension files to the device (first use only)" on page 107.

To upload an application to a BREW 2.x device:

1. Use the data cable that the handset manufacturer provides to attach the handset to a COM port on the computer on which your BREW application files are stored.
2. Navigate to the folder that contains your application files and verify that the directory structure is correct for your target device.

   For complete information about the structure for BREW 2.x devices, see "Device file structures for different BREW versions" on page 97.
3. Select Start > Programs > BREW Tools Suite <*latest version*> > BREW Apploader to start the AppLoader tool. The AppLoader tool displays a connection window.

4. Select the number of the COM port your device is connected to (if you are unsure what port to select, see the following note), select 2.x as the BREW version for your device, and then click OK to connect to the device.

NOTE	You can find the COM port your device is connected to in the Windows Device Manager by looking at the properties of the port that corresponds to the installed device. If you always connect the same device model to the same COM port, this number does not change. Because it changes when you use a different device model, you should always check the COM port number when you connect a new device model.

   After you are connected, the AppLoader tool displays the device's BREW file system.

5. To upload your application files (MOD, SIG, and resource files) to the handset, do one of the following:
   - Select File > Directory > New Directory to create a folder with the same name as your application, and then copy your application files to it. See "Device file structures for different BREW versions" on page 97 for details.
   - Drag your application folder from the Windows Explorer to the AppLoader window.
6. Restart the device.

   The device must be restarted before you can run your application.To restart the device, do one of the following:
   - Select Device > Reset from the BREW AppLoader menu
   - Press and hold the End key on the handset.
7. Navigate to Get it Now > Get Going and select the name of your application to start it in the BREW environment.

Uploading applications to a BREW 3.x device

The instructions that follow explain how to upload a Flash Lite application for BREW to an LG VX9800, which includes the BREW version 3.x platform.

This procedure assumes that you have already installed the appropriate USB drivers for your device on your computer. For more information, see "Setting up your system for BREW" on page 87

This procedure also assumes that you have already uploaded the required Flash Lite Extension and Flash Lite Player files to the device. This needs to be done only once, when you first begin using the device. For more information, see "Uploading Flash Lite Extension files to the device (first use only)" on page 107.

To upload an application to a BREW 3.x device:

1. Use the cable that the handset manufacturer provides to attach the handset to a COM port on the computer on which your BREW application files are stored.
2. Navigate to the folder that contains your application files and verify that the directory structure is correct for your target device.

   For complete information about the structure for BREW 3.x devices, see "Device file structures for different BREW versions" on page 97.
3. Select Start > Programs > BREW Tools Suite <*latest version*> > BREW Apploader to start the AppLoader tool.
4. The AppLoader tool displays a connection window.

   Select the number of the COM port your device is connected to, select 3.x as the BREW version for your device, and then click OK to connect to the device.

NOTE	You can find the COM port that your device is connected to in the Windows Device Manager by looking at the properties of the port that corresponds to the installed device. If you always connect the same device model to the same COM port, this number does not change. Because it changes when you use a different device model, you should always check the COM port number when you connect a new device model.

   After you are connected, the AppLoader tool displays the device's BREW file system.
5. To upload your application files (MOD, SIG and resource files) to the handset, do one of the following:
   - Select File > New > Directory > New Directory to create a directory with the same name as your application, and then copy your application files to it. See "Device file structures for different BREW versions" on page 97 for details.
   - Drag your application folder from the Windows Explorer to the AppLoader window.

6. To restart the device, do one of the following:
   - Select Device > Reset from the BREW AppLoader menu
   - Press and hold the End key on the handset.
7. Navigate to Get it Now > Get Going and select the name of your application to start it in the BREW environment.

   On some devices, your application will not be displayed in the standard Get Going menu. To display your application on these devices, follow these steps:

   a. On the Verizon home screen, press Select/OK to access the menu.
   b. Press the zero key (0) to access the service menu.
   c. Enter the default password, which is six zeros (000000).
   d. Press the nine key (9) to display Get it Now.
   e. Press the g key to access the screen on which Flash Lite applications are displayed.

Testing applications on the device

Before you can start and test your application on the device, you must have a test signature from QUALCOMM. Authenticated developers can use a Web-based tool called Testsig on the BREW Developer extranet to generate test signatures. For specific information on how to generate and use your Testsig, see the BREW website. You can find extensive information on how to test your BREW applets to ensure that they ultimately comply with TBT (True BREW Testing) standards on the BREW Developer extranet.

Where to find more information

For further information about the AppLoader tool, consult the following sources:

Document name and type	Location
Starting with BREW, "About the BREW Tools Suite" (PDF)	Go to the BREW website and then select Developer Home › Application Developer Documentation
BREW)_Apploader.chm (Help)	C:\PRogram Files\BREW Tools Suite ‹*latest version*›\BREWApploader (default)
BREW Testing generator	Go to the BREW website and navigate to the SDK download page

CHAPTER 6

Optimizing Content for Performance and Memory

6

To optimize your Flash content, you must pay attention to basic principles. For example, Flash developers have often had to avoid extremely complex artwork, excessive tweening, and overusing transparency.

Although earlier versions of Flash have resolved most of these performance issues for the desktop, Adobe Flash Lite developers still face additional challenges due to limitations of mobile devices: some devices perform better than others, sometimes dramatically, and because mobile authoring often requires publishing to many different devices, developers must sometimes author for the lowest common denominator.

Optimizing mobile content often requires making trade-offs. For example, one technique might look better, while another results in better performance. As you measure these trade-offs, you will be going back and forth repeatedly between testing in the emulator and testing on the target device.

In Flash 8, the emulator for Flash Lite 2.x was part of the Flash authoring environment. In Flash CS3, the emulator functionality is part of Adobe® Device Central CS3. Device Central lets you emulate your Flash Lite projects on a variety of devices, and can emulate device display, memory use, and performance on specific devices. For complete information about using the emulator to optimize your Flash Lite content for mobile devices, see “Best practices for content on mobile devices” in the Device Central documentation.

CHAPTER 7

Testing Flash Lite Content

7

Adobe Flash CS3 Professional includes an Adobe Flash Lite emulator available on Adobe Device Central CS3 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 CS3 Professional.

This chapter contains the following topics:

Overview of Flash Lite testing features

The Flash Lite testing features in Flash CS3 Professional are part of Adobe Device Central, which includes both an extensive database of device profiles and a device emulator. Device Central also works with many other Adobe products, such as Adobe Creative Suite® and Adobe Dreamweaver.®

The Adobe Device Central emulator lets you preview your Flash Lite content within the Flash authoring tool. The emulator is configured to match the behavior and appearance of the target device. You use the Device Central interface to select and manage your target devices, as well as to specify your application's target Flash Lite content type, such as ring tone, browser, or stand-alone application. In Device Central, 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. For more information about the Flash Lite content types, see "About content types" on page 120.

See the Device Central help for more information, including details about interacting with the emulator.

Testing features not supported by the emulator

The Adobe Device Central emulator does not support all the features available in the standard (desktop) test window. The following is a list of features available in the desktop Flash test window that are not available in the Adobe Device Central 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 > Toolbars > Controller)

Testing inline text in Flash Lite 2.1

You cannot currently test the inline text functionality in the Adobe Device Central emulator; you must test this feature on a device. When testing in the emulator, you must edit the contents of input text fields using a modal dialog box that appears over the Flash Lite content (that is, the emulator functions the same way it did for Flash Lite 1.x and Flash Lite 2.x). For details on how the modal dialog box works, see "Using input text fields" on page 39.

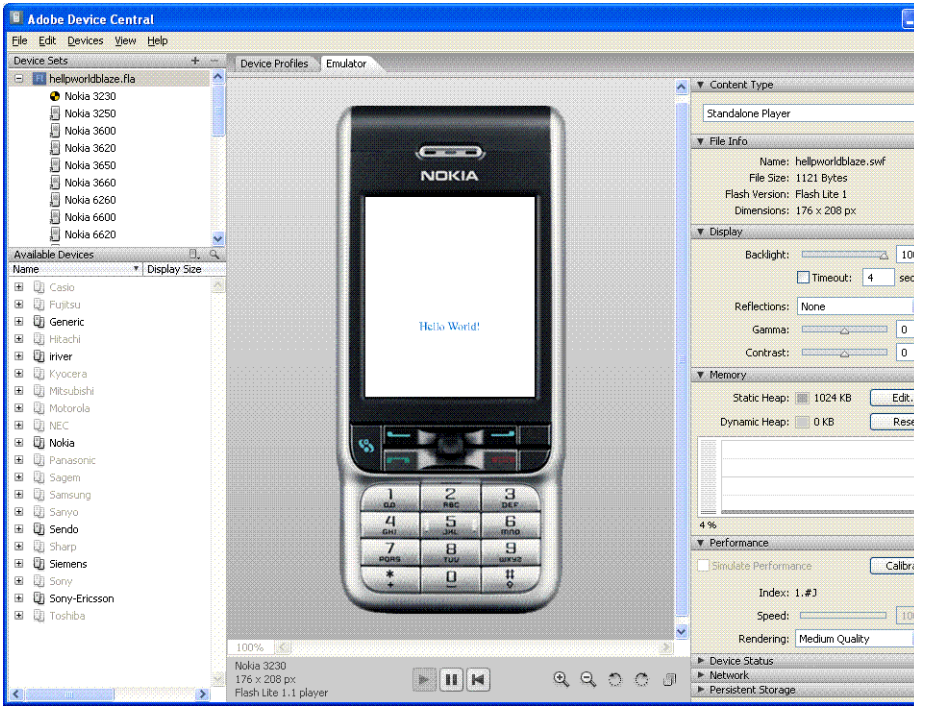
Using the emulator

To start the emulator, choose Control > Test Movie in the Flash authoring tool, the same way you preview your Flash desktop content. However, Adobe Device Central has a different appearance and functionality than the test window for Flash desktop content.

This section contains the following topics:

Invoking Adobe Device Central

After you choose Control > Test Movie or press Ctrl+Enter to start Device Central, a progress bar appears to indicate that Flash is exporting your SWF to Device Central. Once the export is complete, Device Central launches with the focus on the emulator, and loads your SWF:



In Device Central, the Device Sets list shows all target devices that Flash saved with your application. By default, the first device in the set is selected for emulation. The content type selected is the content type that was saved with the application when you created it (for details about the Flash Lite content types, see "About content types" on page 120).

You can test the application with a different content type and different devices. Changing the content type and adding or removing devices from the device set automatically changes the device settings in Flash.

To test how the content appears on a different device, double-click another device in either the top or bottom list. A spinning icon appears next to the device you are testing, and the emulator changes to display your application running in the device you selected.

Setting emulator debug options

The Adobe Device Central emulator can send debugging messages to the Flash Output panel while content is running; the emulator also displays a pop-up version of the Output panel showing the same messages.

The emulator reports the following types of information:

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 2.x ActionScript Language Reference*.

Information messages that contain general information about the selected test device, SWF file size, and other information. These messages appear in the emulator's Alert panel.

Warning messages that contain information about problems with your Flash Lite content that might affect playback.

You can filter the type of information that the emulator generates as follows.

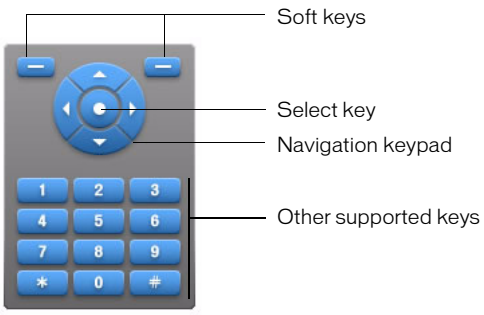
To set Flash Lite output options:

1. Select Control > Test Movie. Flash exports your application to Adobe Device Central and displays it in the emulator.
2. Select View > Flash Output > Show in Device Central.
   - Select or deselect the Trace option.
   - Select or deselect the Information option.
   - Select or deselect the Warnings option.

Interacting with the emulator

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

- Number keys (0 to 9), and the asterisk (*) and pound (#) keys
- Navigation keypad (left, right, down, up, and select)
- Left and right soft keys



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

- The arrow keys on your keyboard (left, right, up, down) map to the corresponding navigation keys on the emulator's navigation keypad.
- The Enter or Return key corresponds to the emulator's select key.
- The Page Up and Page Down keys correspond to the emulator's left and right soft keys, respectively.
- The number keys on your keyboard map to the corresponding number keys on the emulator keypad.

For additional details about interacting with the emulator to test your application, see the Adobe Device Central online help.

About content types

Flash Lite is installed on a variety of devices. Each 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. Other devices use Flash Lite to render Flash content that is embedded in mobile web pages.

The following table lists and describes all of the Flash Lite content types available as of this writing. For additional and up-to-date information about Flash Lite content-type availability, see the Flash Enabled Mobile Device page at www.adobe.com/mobile/supported_devices/.

NOTE	As of this writing, Flash Lite 2.0 supports the Standalone Player content type only.

Flash Lite supports the following content types:

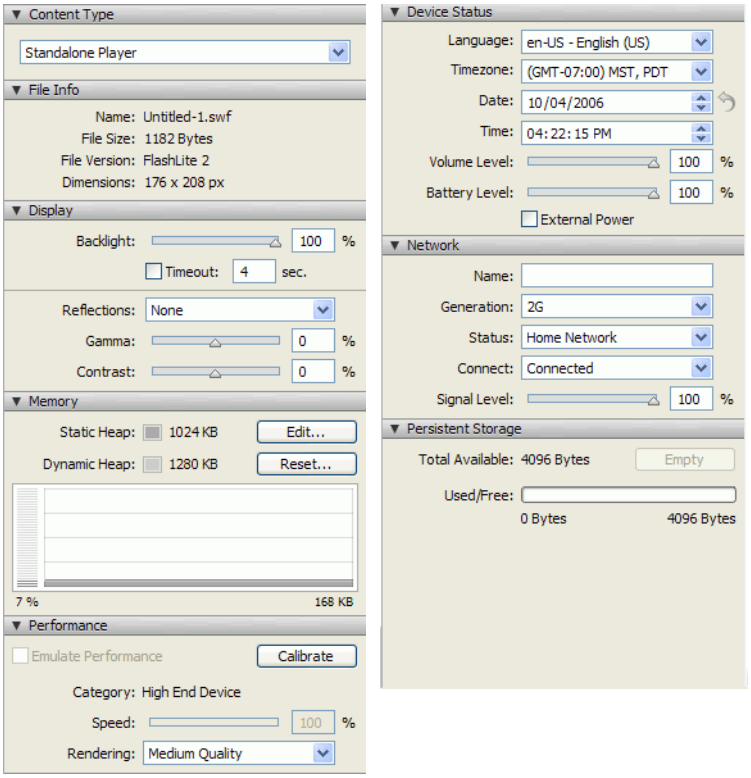
Flash Lite content type	Description	Availability
Address Book	Uses Flash Lite to let users associate a SWF file with an entry in their device's address book application.	DoCoMo and VodafoneKK (Japan only)
Alarm	Uses Flash Lite to let the user select a SWF file to play for the device's alarm.	KDDI and VodafoneKK (Japan only)
Browser	Uses Flash Lite to render Flash content embedded in mobile web pages and viewed in the device's web browser.	DoCoMo, KDDI, and VodafoneKK (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 the 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.	DoCoMo and KDDI (Japan only)
Chaku Flash	Uses Flash Lite to let the user select a SWF file to play as the ring tone for incoming calls.	KDDI (Japan only)

Flash Lite content type	Description	Availability
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.	DoCoMo, KDDI, and VodafoneKK (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 application (used by the UILauncher content type).	KDDI (Casio phones only)
Image Viewer	Use the Image Viewer application that lets the user manage and preview multimedia files on the device, including SWF files.	DoCoMo (Japan only)
Incoming Call	Uses Flash Lite to display an animation when the user receives a call.	DoCoMo, KDDI, and VodafoneKK (Japan only)
Mailer	Uses Flash Lite to display an animation when the user sends or receives an e-mail message.	VodafoneKK (Japan only)
Multimedia	Uses Flash Lite to preview SWF files (as well as other multimedia formats).	KDDI (Japan only)
My Picture	Uses the My Picture application that lets the user manage and preview SWF files on the device, as well as other image formats.	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)

Flash Lite content type	Description	Availability
Screen Saver	Uses Flash Lite to display the device's screen saver.	KDDI and VodafoneKK (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 start 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).	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)
UILauncher	Uses Flash Lite for the device's application launcher.	Uses Flash Lite to display the device's launcher application (that is, the application that lets the user start other applications).
Wake Up Screen	Uses Flash Lite to display an animation as the phone is starting.	DoCoMo (Japan only)

Flash Lite specific information in the emulator

The emulator includes panels that supply information specific to your Flash Lite application. The panels appear along the right side of the window; you can collapse and expand them as you do in Flash.



- The Content Type panel shows the default content type for your application and allows you to select other applicable content types.
- The File Info panel displays the filename, Flash file version, dimensions, and size in kilobytes.
- The Alert panel appears when there is a problem and displays warning messages.
- The Memory and Performance panels show the size of static and dynamic memory and allow you to adjust various parameters to tune your application for performance.
- The Device Status panel shows the values of various platform-specific (`fscommand()`) settings, such as time zone and battery level.
- The Network panel displays information about your network connectivity.

- The Persistent Storage panel shows you how much storage on the device has been used. This is a per-device value: if multiple applications write to the persistent storage, the value is the sum of all their data. You can click Empty to clear the store for that particular device, in order to remove the persistent objects for all content that ran on that device.

For details about these panels and how to use them, see the Adobe Device Central online help.

About screen size and available Stage size

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 can be equal to, or less than, the device's full screen size.

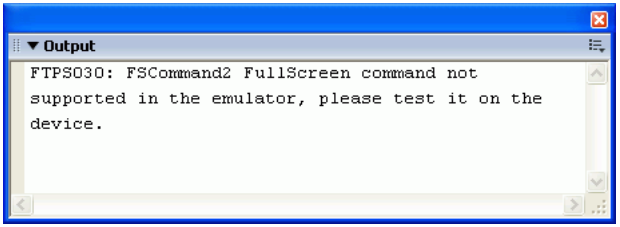
For example, the Stage area that is available to a SWF file running in full-screen mode in the stand-alone player on a Nokia Series 60 device is equal to the device's full screen size (176 x 208 pixels). On other devices (such as those available in Japan), the Stage area that is available to a SWF file running in one of the specialized content types (such as Address Book or Screensaver) might be less than the device's total screen size. For example, the Fujitsu 700i has a screen size of 240 x 320 pixels; 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 actual dimensions are different from the available Stage dimensions, the Flash Lite player scales the content (proportionately) to fit within the available Stage area. When you test your content in the emulator, the emulator also warns if your application's Stage size is different from the available Stage area.

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

Flash Lite features not supported by the emulator

The emulator doesn't support all the ActionScript commands and player 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 Short Message Service (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:



You must test your SWF file on an actual device to confirm that those features are functioning as expected.

Unsupported ActionScript commands

The Adobe Device Central emulator does not support the following `fscommand()` and `fscommand2()` commands:

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

About playing a device video in the emulator

The Flash Lite player uses the device's default video handler application to play video content in your SWF file, rather than decoding the video natively. This practice lets Flash Lite developers use any video format that the target device supports, such as 3GPP, MPEG, or AVI. For more information about using video in Flash Lite, see "Using device video" on page 68.

The Adobe Device Central emulator uses QuickTime Player to render device video when testing in the Flash authoring tool. The latest version of QuickTime Player (version 7 as of this writing) supports playback of several different device video formats, including 3GPP and others. However, by default, QuickTime might not support some video formats during playback that an actual device supports, and therefore those formats will not play in the Adobe Device Central emulator. For this reason, it's important to always test your content on an actual device.

If your device video does not play in the QuickTime Player, by default, try the following:

- Upgrade to the latest version of QuickTime Player.
- If available, install a third-party video codec (short for compressor-decompressor) that supports the video format you're using.

Errors

The Adobe Device Central emulator can generate alerts as you test your content. One type of alert appears only in the emulator and is intended to provide information about actual or potential errors; the other type of alert occurs in both the emulator and on an actual device.

The first type of alert provides debugging information about your SWF file. For example, if your SWF file 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 an alert.

The other type of 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 of 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 is 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 file 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	N/A	N/A
6	Bad image data	The SWF file contains an image that either Flash Lite, or the platform's native image decoder, was unable to decode.
7	Bad sound data	The SWF file attempted to load a sound in an unsupported format, or the sound data is corrupted.
8	Root movie unloaded	This error occurs when the root (level 0) SWF file is replaced with another SWF file.
9	N/A	N/A
10	getURL string too long	The URL string in the getURL call is too long.
11	Not enough memory to decompress image.	Flash Lite does not have enough memory to decode an image in the SWF file.

Error number	Error string	Description and possible causes
12	Bad SVG data	Flash Lite attempted to load SVG data that is corrupted.
13	Stream loading out of memory	Flash Lite does not have enough memory available to handle the data being streamed from a URL. For example, this error can occur if you attempt to load an XML file over the network that's too large for Flash Lite to handle at one time. If possible, try breaking the data file into several smaller files and then load each file individually.

APPENDIX A

Warning and Error Messages

A

This appendix lists the possible information and warning messages that the emulator in Adobe Device Central might generate while you're testing a Flash Lite application. Messages appear in a pop-up Output window in the emulator and Flash also reports them in its Output panel.

For lists of errors that can occur on both the emulator and an actual device, see "Errors" on page 127.

Adobe Device Central emulator error and information messages

The following table lists all the information messages that the Adobe Device Central 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.

Error code	Message	Description
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.
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 process only `getURL()` function calls that result from users 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 the *Flash Lite 2.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.

Error code	Message	Description
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 process only `loadMovie()` function calls that result from users pressing a key on their device. The emulator detected that your application made a call to `loadMovie()` that wasn't associated with a keypress.
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 process only `loadVariables()` function calls that result from users 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.

Error code	Message	Description
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 process only `fscommand()` function calls that result from users 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 do 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.
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 users pressing a key on their device. (This restriction does not apply to Flash Lite 2.0.)
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.

Error code	Message	Description
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 Adobe Device Central 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 `fscommand2 function` in the *Flash Lite 2.x ActionScript Language Reference*.
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 whether the specified command functions as expected,

Error code	Message	Description
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 run-time restrictions when played on the selected device. Test your SWF file on an actual device to see whether 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 run-time 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 run-time 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.
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.

Error code	Message	Description
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 "Default navigation modes" on page 16.
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 "Default navigation modes" on page 16.
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.

Error code	Message	Description
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.
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 an 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 an 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 an 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 an 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 an 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 an 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 played.

Error code	Message	Description
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.
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" on page 32.
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 whether the specified command functions as expected.
FTPS110	Soft keys are not supported in the Flash Lite 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

E

F

G

H

I

K

L

M

N

P

R

S

T

U

V

W

X