Developing Applications for Adobe® Flash® Lite® for the Digital Home
## Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is Flash Lite for the digital home?</td>
<td>1</td>
</tr>
<tr>
<td>Potential of the Adobe Flash Platform on TV</td>
<td>1</td>
</tr>
<tr>
<td>Documentation map for authors</td>
<td>1</td>
</tr>
<tr>
<td>Your authoring environment</td>
<td>2</td>
</tr>
<tr>
<td>Handling video</td>
<td>2</td>
</tr>
<tr>
<td>Handling user interaction and focus</td>
<td>4</td>
</tr>
<tr>
<td>Handling graphics</td>
<td>7</td>
</tr>
<tr>
<td>Handling audio</td>
<td>8</td>
</tr>
<tr>
<td>Additional topics</td>
<td>8</td>
</tr>
<tr>
<td>ActionScript API differences</td>
<td>9</td>
</tr>
<tr>
<td>Error handling and debugging</td>
<td>10</td>
</tr>
</tbody>
</table>
What is Flash Lite for the digital home?

Adobe® Flash® Lite® for the digital home is Flash® Player optimized for the hardware and software architecture of digital home electronics, such as television sets and Blu-ray players. You, the Adobe® Flash® developer, can create applications for Flash Lite for the digital home that stream and play high-definition video from the Internet. You can also create rich Internet applications for Flash Lite for the digital home.

Flash Lite for the digital home is like Flash Player 8 rendering combined with Flash Player 10 media. For example, Flash Lite for the digital home handles media and data just as Flash Player 8 does: using NetStream and NetConnection to open a stream to the Internet and delivering media over HTTP and HTTPS (or, if you use Adobe® Flash® Media Server, over RTMP and RTMPE). Video playback occurs through a Video object on the Stage. Flash Lite for the digital home offers selected advanced capabilities of Flash Player 10, such as support for hardware playback of H.264 video and HE-AAC audio.

Potential of the Adobe Flash Platform on TV

The Adobe Flash Platform has long been recognized as a leader in delivering high-quality web video and providing rich content-driven user experiences. Applications built using Flash Lite for the digital home have enormous potential to deliver video content from the Internet to home television sets. Flash Lite for the digital home uses both hardware acceleration and software acceleration to deliver a first-class experience for users with high-speed bandwidth. The ubiquity and reliability of the Flash Platform lets content distribution networks, broadcasters, and producers of all sizes implement their applications on a unified runtime model: use the Flash Platform to deliver on-demand streaming video over the Internet to PCs, mobile devices, and, now, to television sets at home as well.

Applications built on the Flash Platform also excel in providing rich interactive user experiences. With Flash Lite for the digital home, developers can create games, social networking applications, and real-time, data-driven applications such as weather or stock checker applications. These applications can run on Internet-connected home televisions, set-top boxes, Blu-ray players, and other devices in the digital living room.

With the tight integration that is available between Flash Lite for the digital home and the home electronic devices on which it runs, Flash developers can even create complete interfaces to control those devices, such as program guides or interfaces to schedule recording and playback.

Documentation map for authors

This document is intended for the experienced Adobe® Flash® Lite® developer who has a strong understanding of Adobe® ActionScript®. It describes the ways in which application development for Flash Lite for the digital home differs from development for Flash Lite 3.1. The document focuses on the differences between Flash Lite for the digital home and Flash Lite 3.1, as well as the special factors to consider when developing for digital home-electronic devices.

The following documentation map describes the different documents to which you may want to refer, depending on the information you need:
**Your authoring environment**

Use Adobe® Flash® CS4 Professional or Adobe® Flash® CS3 Professional as your authoring environment. Flash Lite for the digital home does not support applications developed in Adobe® Flex®. Use ActionScript 2.0 in your applications.

In Flash CS4 Professional or Flash CS3 Professional, in your publish profile, set your target ActionScript version to ActionScript 2.0. Set your target version of Flash Player to Flash Player 8 or Flash Lite 3.1. However, if you are using the ActionScript 2.0 NetStream methods for dynamic streaming, use Flash CS4 Professional. Set your target version of Flash Player to Flash Player 10. For more information, see “Using dynamic streaming” on page 3.

In the Actions Panel, select either ActionScript 1.0 & 2.0 or Flash Lite 3.1 ActionScript. Keep in mind that certain APIs listed are not supported by Flash Lite for the digital home; see “ActionScript API differences” on page 9.

For full-screen applications, set your Stage size to 1280 pixels x 720 pixels.

**Handling video**

Flash Lite for the digital home handles media and data just as Flash 8 does: using NetStream and NetConnection to open a stream to the Internet and delivering media over HTTP and HTTPS (or, if you use Adobe® Flash® Media Server, over RTMP or RTMPE). Video playback occurs through a Video object on the Stage.

On the Flash Lite for the digital home platform, high definition 1080p H.264 video playback is fully supported. H.264 video is decoded, rendered, and played back on the hardware. Because hardware support for H.264 video is universal among television devices, Adobe recommends that you use H.264 video when practical. This practice allows you to take advantage of hardware video functionality.

Because H.264 video is rendered through the hardware, some limitations with H.264 video transformations exist. You cannot stretch, skew, flip, or rotate H.264 video in Flash. You cannot use a transparency value on a video to blend with objects behind the video. Filters are not supported. Video can only be displayed in a rectangular region. You can, however, resize and move video while it plays and blend objects on top of a video.

Other video formats common in Flash, such as On2 and Sorenson, are also supported by Flash Lite for the digital home. However, these formats are typically decoded and rendered through the software. Hardware decoding and rendering of On2 and Sorenson video is device-dependent and not as common as for H.264 video. When software in Flash Lite for the digital home renders these types of video, you can transform video (rotate, skew, and so on).

**See also**

*Working with video in Flash Lite*
Graphics overlays and hardware video

In applications for Flash Lite for the digital home, video decoded and rendered on the hardware is always rendered in the bottom-most graphics plane. You cannot set transparency on the video plane and move movie clips behind video. When movie clips are animated under the Video object, they disappear as if the Video object is fully opaque.

Movie clips that are animated on top of the Video object behave as expected.

Designing for the safe viewing area

When you design a full-screen application for digital home-electronic devices, consider overscan and the safe viewing area. Overscan is the portion of the actual image area that falls outside the actual viewing area of a screen. The area that is not overscanned is the safe viewing area. The safe viewing area can vary depending on the device and the way the viewer has configured their television.

With regards to overscan on digital home-electronic devices, the design of interfaces and rich Internet applications in Flash Lite requires special consideration. Is your application designed to fill the entire television screen? If so, avoid putting critical elements of your application — such as text, buttons, or graphics — within approximately 7.5% of each edge of the Stage. This practice ensures that your entire SWF application is visible on most television screens and that edges are not likely to be cropped unexpectedly.

The following image shows the safe viewing area for a full-screen application:

Using dynamic streaming

Flash Lite for the digital home supports dynamic streaming. You can use dynamic streaming when Flash Media Server is delivering a video stream to your application. Your application can send commands to Flash Media Server to switch between versions of a content stream that are encoded at different bit rates. The transition occurs seamlessly in your application when the content streams use the same codec type. You use ActionScript 2.0 to send the commands to switch streams.

Switching among different versions of the content stream lets your application adapt to changing network conditions that affect the user’s actual available bandwidth. The result is a smooth and uninterrupted viewing experience for the user.
For an overview of dynamic streaming, see the following documentation:

- **Dynamic Streaming** in *Adobe Flash Media Server 3.5 Developer Guide*, available in Flash Media Server Help.

- **Dynamic streaming in Flash Media Server 3.5 -- Part 1: Overview of the new capabilities.** This article is available on the Adobe Developer Connection website www.adobe.com/devnet, in the Flash Media Server Developer Center.

- **Live dynamic streaming with Flash Media Server 3.5.** This article outlines Adobe’s recommendations for developing and enhancing the performance of dynamic streaming with Adobe Flash Media Server 3.5. This article is available on the Adobe Developer Connection website www.adobe.com/devnet, in the Flash Media Server Developer Center.

Flash Lite for the digital home supports the same ActionScript 2.0 NetStream functionality as Flash Player 10. Use the following ActionScript 2.0 methods to implement dynamic streaming:

- `NetStream.play2()`
- `NetStream.getInfo()`
- `NetStream.onStatus()`
- `NetStream.setBufferTime()`

For detailed information about ActionScript 2.0 for dynamic streaming, see the following documentation:

- **ActionScript guide to dynamic streaming.** This article is available on the Adobe Developer Connection website www.adobe.com/devnet, in the Flash Media Server Developer Center.

- **Adobe ActionScript 2.0 Language Reference for Adobe Flash Media Server 3.5**, available in Flash Media Server Help.

For high definition video, Adobe recommends you use `NetStream.setBufferTime()` to set the buffer time to 5 seconds or less. A longer buffer time requires more memory for the video buffer. Furthermore, the high bit rate of high definition video requires more memory per second of video than standard video. However, less memory is available in systems running Flash Lite for the digital home than in PCs. Therefore, a buffer time greater than 5 seconds typically uses too much memory for Flash Lite for the digital home systems.

**Note:** Some of the referenced documents discuss the ActionScript 3.0 version of NetStream. However, the ActionScript 2.0 version of NetStream has similar APIs. Therefore, the documents do apply to Flash Lite for the digital home. Also, some of the documents discuss the ActionScript 3.0 DynamicStream class, which extends the NetStream class. ActionScript 2.0 does not support the DynamicStream class. In ActionScript 2.0, you can use the DynamicStream class as a reference to write your own dynamic streaming algorithms. You can download the DynamicStream class from Adobe Flash Media Server productivity tools and sample downloads. These tools are available at www.adobe.com.

**Handling user interaction and focus**

Applications for Flash Lite for the digital home typically require an interface driven only by keys (no pointer device). Selection state in a key-driven interface does not include activation, unlike clicking a mouse, which both selects and activates a user interface element. In a key-driven interface, selecting and activating (that is, clicking Select or OK) are two separate steps.

Remote controls are the primary user interface mechanism and differ between hardware manufacturers. Flash Lite for the digital home supports four-way navigation and key-driven events using the remote control’s directional keys and Select button to navigate an application, change focus between objects on the screen, and activate or select a function or operation.
**Key handling**

The following table lists the key codes and ActionScript Key class constants for the main navigation and selection keys for an application:

<table>
<thead>
<tr>
<th>Description</th>
<th>Key code</th>
<th>ActionScript Key class constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directional keys: up, down, left, right</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Up: 38</td>
<td></td>
<td>Key.UP</td>
</tr>
<tr>
<td>Down: 40</td>
<td></td>
<td>Key.DOWN</td>
</tr>
<tr>
<td>Left: 37</td>
<td></td>
<td>Key.LEFT</td>
</tr>
<tr>
<td>Right: 39</td>
<td></td>
<td>Key.RIGHT</td>
</tr>
<tr>
<td>OK or Select</td>
<td>13</td>
<td>Key.ENTER</td>
</tr>
</tbody>
</table>

Additional keys are supported. Adobe provides an ActionScript file that maps key constants used in remote controls to ActionScript; to get this file, contact your Adobe business partner.

To ensure that the most common remote control keys can be handled with ActionScript, Adobe provides a default key mapping file to hardware vendors. However, be aware that hardware vendors can change or add new key codes to the key definition file. It can be helpful to get all applicable key codes for a particular device using the `Key.getCode()` function.

For any keys that do not map to ActionScript 2.0 constants, the `Key.getAscii()` function returns a value of 0. The ASCII value for the numeric keys is ‘0’ (0x30) through ‘9’ (0x39).

The following example demonstrates getting and using key values to control navigation in a video player application. This code snippet uses `Key.getCode()` to map the pressing of certain keys to ActionScript functions.

The object `flvplayer` is an instance of the FLVPlayback component on the Stage. To enable use of the remote control’s directional keys for seeking in a video, focus and tabbing on the FLVPlayback component are disabled. The OK button is mapped to `play()` and `pause()` functions.
//To use directional keys for seeking, disable focus and tabbing on FLVPlayback component
flvplayer.focusEnabled = false;
flvplayer.tabEnabled = false;
flvplayer.tabChildren = false;
flvplayer.contentPath = "http://urlToVideo/someVideo.mp4";
var myListener:Object = new Object();
myListener.onKeyDown = function() {
    // Toggle play/pause using "OK" button
    if (Key.getCode() == Key.ENTER) {
        if (flvplayer.playing) {
            flvplayer.pause();
        } else {
            flvplayer.play();
        }
    }
    // Seek Previous using Left directional button
    if (Key.getCode() == Key.LEFT) {
        flvplayer.seekPercent(flvplayer.playheadPercentage-5);
        flvplayer.play();
    }
    // Seek Next using Right directional button
    if (Key.getCode() == Key.RIGHT) {
        flvplayer.seekPercent(flvplayer.playheadPercentage+5);
        flvplayer.play();
    }
};
Key.addListener(myListener);

Understanding default navigation

Flash Lite for the digital home supports four-way navigation, without wraparound. Pressing a navigation key moves the focus to the nearest object that can receive focus (such as a button or text field) located in that direction, if one exists. The nearest focusable object is determined by measuring the distance from the center of the current object to the center of all other objects. If no focusable object exists in that direction, then the focus does not change from the current object.

To ensure that the navigation order behaves as expected, follow these guidelines. These guidelines are applicable to user interface elements such as a menu constructed of buttons or text fields.

- Avoid setting the autoSize property to true for text fields, because auto-sizing changes the center point of a text field.
- Make each button or text field a constant width.
- Align all objects on the Stage. Say that you have a vertical navigation bar with eight buttons, all equal width. Align all the buttons in the bar so that focus moves predictably from one button to the next or previous.

Text entry

Digital home-electronic devices that permit text entry or support text entry with an external keyboard are currently rare. To support text entry, your primary option is to create a graphical keyboard in Flash, displayed on the TV screen, that a user navigates with the directional keys. To enter text in a text field, remember that, after the text field receives focus, the user must click OK or Select before inputting text.

A user can enter numbers with the number keys on a remote control.
Handling graphics

Flash Lite for the digital home supports PNG and JPG graphics, and supports hardware acceleration for both.

**Designing for native hardware acceleration**

Flash Lite for the digital home includes a hardware acceleration feature for graphics. You can take advantage of increased performance by minimizing the use of the following elements in your application:

- Non-rectangular shapes.
- Rectangular shapes that are smaller than 2 x 4 pixels.
- Bitmaps or rectangular shapes with the same fill color or texture that have been combined into a single object.
- Shapes with a gradient fill.
- Shapes with color maps that are not composed of an alpha-only blend.
- Vector graphics.
- On some platforms, scaling bitmaps can negatively affect performance.

Use PNGs and JPGs for specific purposes:

- Use PNG for buttons and other static images that are part of the user interface.
- Use JPG for other images, such as photographs or artwork. Keep in mind that lower-quality JPG files can have artifacts that become apparent on large-screen television sets. JPG files are more likely to be hardware-accelerated than PNG files.
- The trade-off between using PNG and JPG is image quality and performance.

Finally, consider that the order in which you load different kinds of images matters. When possible, load all hardware-accelerated images first, then load any vector graphics. For example, load all JPG graphics in your application, then load text which is a vector graphic.

**Bitmaps, vector graphics, and the cacheAsBitmap property**

Flash Lite for the digital home supports bitmap caching of movie clips. Caching a movie clip as a bitmap prevents Flash Player from having to continually redraw the image, which can provide a significant improvement in playback performance.

Use bitmap caching on complex vector graphics to optimize performance. Bitmap caching is best used on movie clips that have mostly static content and that do not scale or rotate. You can set the `MovieClip.cacheAsBitmap` property to `true` or turn on bitmap caching in Flash CS4 Professional by selecting the Use Runtime Bitmap Caching option in the Property inspector. For more information on bitmap caching and when to use it, see the Flash CS4 documentation.

**See also**

- [About runtime bitmap caching](#)
- [MovieClip.cacheAsBitmap property](#)
Transforming bitmaps

Flash Lite for the digital home provides the capability to use hardware acceleration for scaling, flipping, and rotating bitmaps. However, availability of hardware acceleration depends on the digital-home hardware platform. Because hardware acceleration increases application performance, many digital-home hardware platforms support hardware acceleration for all or some of bitmap scaling, flipping, and rotating.

Handling audio

Support for audio-only streams is either native to Flash Lite or is decoded and played through the device hardware (called hardware audio). The distinction between native audio and hardware audio is important to understand. Different devices support different combinations of simultaneous playback of native audio and hardware audio.

Note: Device audio as implemented in Flash Lite 3.x is not implemented in Flash Lite for the digital home.

Native audio includes event sounds and stream sounds. Native audio is decoded by Flash Lite and played through a Sound object. Typically, this category includes short and simple sound effects or background music, such as mp3 files that are not wrapped in an FLV container.

Depending on the hardware implementation, certain kinds of audio can be decoded and played by the device hardware (hardware audio). The data source is an FLV or F4V container and is played through the Video object.

Keep in mind the following issues when working with different types of audio:

- Native audio can play simultaneously with hardware audio, but this functionality depends on the device hardware and is not guaranteed to work.
- Depending on the hardware, in an application with multiple SWF files running simultaneously, native audio from only one SWF file can be played at a time.
- To play audio-only HE-AAC streams, wrap them in an F4V or FLV container and play them using a Video object, instead of a Sound object.

You can play multiple native stream sounds simultaneously. However, you can play only one Flash Lite native event sound at a time. Event sounds cannot be played simultaneously with stream sounds. If there is a conflict between event and stream sounds, the event sounds play. These restrictions are similar to Flash Lite 3.x.

Additional topics

Security considerations

The security model in Flash Lite for the digital home is similar to the model for Flash Lite 3.x, which is based on the Flash Player 8 security model. One difference exists in Flash Lite for the digital home: the local file security sandbox. By default, all local SWF files are placed in the local trusted sandbox. The local trusted sandbox allows local SWF files to read from local data sources, communicate with the Internet, interact with any server, and script other SWF files.

The local trusted sandbox allows your SWF application to deliver video and other data from other websites. The device manufacturer can change this default setting on a global level (that is, for all SWF files that run on that device) or for individual SWF files. The device setting overrides any security setting you make in your publish profile.
Memory considerations
The amount of memory available to Flash Player varies across devices:

![Amount of memory available to Flash Player](image)

The continuum of memory availability across different devices

Digital home-electronic devices generally have more memory available for Flash applications than phones, but less than PCs. Applications designed to work within the relatively limited memory capacity of mobile devices can be expected to work on Flash Lite for the digital home.

User interface design guidelines
Consider the following general guidelines when you design your application:

- Like applications for mobile devices, digital home-electronics applications are key-driven: users press keys to navigate and activate selections.
- Unlike applications for mobile devices, the application is presented to the user on a large screen. Users view the application from a distance, not up close.
  In the digital living room, large font size and good contrast make text easier to read.
- The potential user audience for digital home-electronic applications is far broader than for PCs or mobile phones. The potential user ranges from the technically savvy user to non-technical, non-computer users.

ActionScript API differences
Flash Lite for the digital home contains the ActionScript API functionality that exists in Flash Lite 3.1, with some exceptions. Conversely, Flash Lite for the digital home contains some functionality that doesn’t exist, behaves differently, or was not documented in Flash Lite 3.1.

See the Flash Lite 3.x ActionScript documentation for a complete list of APIs that are available in Flash Lite 3.x.

Unsupported functionality as compared to Flash Lite 3.1
The following ActionScript APIs are supported in Flash Lite 3.1 but are not supported in Flash Lite for the digital home.

- Soft keys (ExtendedKey class and keycodes SOFT1-SOFT12)
- fscommand2()
- Matrix.createGradientBox()
- MovieClip.getUrl(). Use loadMovie() instead.
- MovieClip.beginGradientFill()
- MovieClip.transform()
- TextFormat.target
- TextFormat.url
**Functionality added to Flash Lite for the digital home**

The following APIs are supported in Flash Lite for the digital home. The functionality is either fully equivalent to Flash 8 or has partial functionality as compared to Flash 8. These APIs were either not supported or not documented in Flash Lite 3.1, and they are listed here for your convenience. For more information about these APIs, see the *Flash ActionScript 2.0 Language Reference*.

<table>
<thead>
<tr>
<th>API name</th>
<th>Support in Flash Lite for the digital home</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoadVars.onHTTPStatus</td>
<td>Full</td>
</tr>
<tr>
<td>MovieClip.beginBitmap()</td>
<td>Full</td>
</tr>
<tr>
<td>TextFormat.font</td>
<td>Flash Lite for digital home supports the _sans, _serif, and _typewriter fonts. To use any other font, embed the font in the SWF file. If you publish a SWF file with a device font that is not embedded in the SWF file, Flash Lite maps the font to _sans, _serif, or _typewriter.</td>
</tr>
<tr>
<td>XML.onHTTPStatus</td>
<td>Full</td>
</tr>
</tbody>
</table>

Flash Lite for the digital home also supports dynamic streaming functionality. For more information, see “Using dynamic streaming” on page 3.

**Error handling and debugging**

Adobe recommends that, whenever possible, you use trace statements to debug your code. Output the trace statements to a Linux console for the target hardware platform.