# Using Flash® Media Playback and Strobe Media Playback

Using Flash Media Playback and Strobe Media Playback



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



**Last updated 11/16/2010**

# Contents

# Chapter 1: A Quick Start to Media Playback

Websites use media players to give viewers the rich multimedia experiences they need and expect. An ideal player is full-featured and easy to use, and it should be simple for developers to customize and manage.

Flash® Media Playback and Strobe Media Playback let you focus on the overall user experience of video on your site, not on building and managing media players. Both players are designed for quick and easy deployment. And both provide seamless support for the advanced features of Adobe® Flash® Platform technologies.

**Flash Media Playback** is an out-of-the-box, feature-rich, free media player suitable for designers, content owners, IT professionals, and developers. It is designed to be your simplest deployment solution. You use a setup assistant to configure the player with a few mouse clicks. Because Adobe hosts the player for you, there's nothing for you to install or keep updated. And users experience fast downloads, because the player is stored in their Flash cache.

**Strobe Media Playback** provides more flexibility than Flash Media Playback, while still helping you get up and running quickly. Like Flash Media Playback, the Strobe Media Playback player is both free and easy to customize. Unlike Flash Media Playback, the Strobe Media Playback player is open source, available both as a compiled SWF file and as uncompiled source code. And, because it is a download, it can be deployed behind firewalls, where the Flash Media Playback player might not be accessible.

Both players are based on **Open Source Media Framework (OSMF)**. OSMF is a pure ActionScript® 3.0 framework that gives developers complete flexibility and control in creating their own rich media experiences. For more information on Open Source Media Framework, go to www.osmf.org.

This document discusses both the Flash Media Playback and Strobe Media Playback players. Where something differs between the two, it is noted. Most of the information, though, applies to both.

## Features

The Flash Media Playback and Strobe Media Playback players provide the following:

- A standard appearance (the player's "skin" or "chrome") that is easy to customize.
- Playback for a wide variety of content types, including files of type FLV, SWF, F4V, MOV, MP4, JPG, and MP3; M3U playlists; and F4M metadata manifests.
- Support for both standard and advanced delivery methods, including progressive download, RTMP streaming, RTMP dynamic streaming, HTTP streaming, HTTP dynamic streaming, and live streaming. Flash Media Playback 1.5 and Strobe Media Playback 1.5 also provide support for RTMFP multicast content delivery.
- Automatic management of secure (DRM) content with Flash® Access™.
- Advanced playback, with digital video recorder (DVR) functionality, next/previous track seeking, and playlist navigation.
- Control of capabilities such as autoplay, autohide controls, poster frame definition, control bar positioning, and more, without the use of Flash authoring tools.
- Easy configuration with HTML.

- Simple integration of third-party plug-in services such as CDNs, advertising, and analytics. The flexible architecture gives you the option of compiling plug-ins statically or loading them dynamically, so plug-in providers can perform immediate upgrades and versioning.
- Quality of service (QoS) enhancements, including optimized buffering and dynamic (multi-bitrate) streaming.

# System requirements

Both players support the same operating systems as Flash Player, and each requires Flash Player 10.0 to be installed. (For HTTP dynamic streaming or playing protected content from Flash Access 2.0, you must install Flash Player 10.1.)

The following are the basic system requirements to run the players. See the Adobe Flash Player system requirements for additional information relevant to your system.

| Specification | Minimum | Recommended for High Definition (HD) |
|---|---|---|
| Resolution | 1024x768, 1280x720 | 1920x1080 |
| Processor speed | 1 Ghz | 2 Ghz |
| RAM | 1 GB | 2 GB |
| Video RAM | 128 MB | 512 MB |

# Playing content with Flash Media Playback

Because Adobe hosts the Flash Media Playback player, there is nothing for you to install. The first time Flash Media Playback runs on a system, both the player and a preloader are downloaded automatically. For subsequent uses of the player, only the preloader (approximately 2k in size) is downloaded.

Playing content with Flash Media Playback requires that you provide a small amount of HTML code on your web page. To start, go to the Flash Media Playback site, review the user agreement, then proceed to the setup assistant page.

The setup assistant creates the code that you copy into your web page. It can provide code for a wide variety of options for your player, but you need just two things to begin:

- The location or "source" for the content that you want to play. You provide this information in the form of a fully qualified URL, specifying the complete file hierarchy for where the content is located.
- The size to display the player window. You provide this information with separate values for the window's width and height. Each value is specified either in pixels or as a percentage of the size of the browser window. You may want to include space for the height of the player's control bar (35 pixels) when deciding what height to make the window.

When you supply these values to the setup assistant, it translates your information into HTML code. You can then copy and paste this HTML output into the code for your page. After you load your page, the Flash Media Playback player is ready to go, with its default settings activated and your content loaded.

# Playing content with Strobe Media Playback

Playing content with Strobe Media Playback requires you to place a small amount of HTML code on your web page. However, because the Strobe Media Playback player is not hosted by Adobe, you must first install it on your web server.

## Installing Strobe Media Playback

Follow these instructions to install the player and run its included demo content. Once you have done so, proceed to "Running Strobe Media Playback" on page 3to test your own content on the player.

1. Go to the Strobe Media Playback wiki and review the license information. Strobe Media Playback is an open-source project, licensed under version 1.1 of the Mozilla Public License. For more details, see opensource.adobe.com.
2. Download the Strobe Media Playback zip file from the Strobe Media Playback site.
3. Find the root directory for your web server. It is typically named "htdocs", "html", or "public_html".
4. Create a folder inside the root directory. Open the zip file, and extract the zip contents there.
5. Open your browser and point it to the StrobeMediaPlayback.html demo page that is included in the zip file. Run the demo to confirm that you have successfully installed the player.

## Running Strobe Media Playback

First, read "Installing Strobe Media Playback" on page 3 and install the player. Next, to run Strobe Media Playback, you must create a small amount of required HTML code yourself, then embed this code in your web page. There are three pieces of information that browsers require to run the Strobe Media Playback player:

- A URL providing the location of the content to play.
- A URL for the location where you installed the player.
- Height and width values for the size of the player window to display. You can specify these either in pixels or as a percentage of the size of the browser window. You may want to include space for the height of the player's control bar (35 pixels) when deciding what height to make the window.

You specify each of these using HTML `object` and `embed` tags. (Using both `object` and `embed` is recommended to ensure backward compatibility with older versions of some browsers.)

**For the required `object` tag values:**

1. Provide dimensions for `width` and `height`.
2. In the `movie` parameter, specify a fully qualified (not relative) URL for the location of the player.
3. In the `FlashVars` parameter, use `src` to set the location of the content. This URL can be fully qualified or relative to the location of the player.

That is all that's required. It is recommended that you set `allowFullScreen` to `true`, to take advantage of the player's full-screen capability. And, for both your own debugging purposes and to allow JavaScript access, you may want to set `allowscriptaccess` to `always`.

**For the required `embed` tag values:**

1. Provide dimensions for `width` and `height`.
2. Use the `src` parameter for a fully qualified URL specifying the location of the player. (Note that this syntax differs from that used for the player location with the `object` tag.)
3. In the `FlashVars` parameter, use `src` to set the location of the content.

**4** For strict XHTML compliance, you should set the `type` parameter to `application/x-shockwave-flash`.

As with the `object` tag, you may also want to set `allowfullscreen` to `true` and `allowscriptaccess` to `always`.

You can customize the following HTML code sample for your page. Simply replace the sample values in the code with values for your own content location, player location, and size.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=10,0,0
,0"
width="470" height="320">
    <param  name="movie"
            value="http://myserver.com/strobe/StrobeMediaPlayback.swf"></param>
    <param  name="FlashVars"
            value="src=http://myserver.com/mymovie.flv"></param>
    <param  name="allowFullScreen" value="true"></param>
    <param  name="allowscriptaccess" value="always"></param>
    <embed src="http://myserver.com/strobe/StrobeMediaPlayback.swf"
        type="application/x-shockwave-flash"
        allowscriptaccess="always" allowfullscreen="true"
        width="470" height="320"
        FlashVars="src=http://myserver.com/mymovie.flv">
    </embed>
</object>
```

***Note:*** *URLs for RTMP streams do not typically include file extensions for content. If you are loading content to be played as an RTMP stream, do not use a file extension as part of the URL. Doing so can cause the player to fail to play the content.*

For more information on configuring and customizing your player, see "Configuring the Player" on page 5 and "Changing the Appearance of the Player" on page 12.

For help and troubleshooting tips, visit the Strobe Media Playback developer forum.

# Chapter 2: Configuring the Player

There are two ways you can customize your player. The first is to change how the player works by configuring its features. The second is to change the player's appearance or "chrome". This chapter discusses feature configuration. To learn about customizing the player's chrome, see "Changing the Appearance of the Player" on page 12.

For the Flash Media Playback player, the process of feature configuration is the same as for playing content. The setup assistant produces HTML code that reflects your configuration choices, and you copy and paste that code into your web page. If you are using Flash Media Playback, all you need to know is what your configuration options are and what effects they create.

For Strobe Media Playback, you have several options for creating the code to configure the player, as described in "Configuring Strobe Media Playback" on page 10.

All feature configuration settings are optional. Both players' features are by default set to values that meet the needs of most users.

## Basic configuration options

The following table describes the basic configuration options you can set for your player. For Flash Media Playback, you change these settings with the setup assistant. For Strobe Media Playback, you can set these options as described in "Configuring Strobe Media Playback" on page 10.

| Setting | Name | Possible values | Description |
|---|---|---|---|
| Control bar position | `controlBarMode` | `docked` (default), `floating`, `none` | The location where the player's controls are displayed. The default value sets the controls along the bottom of the player window. A value of `floating` displays the control bar hovering over the content, near the bottom of the window If a value of `none` is set, no control bar is displayed. |
| Control bar visibility | `controlBarAutoHide` | `true` (default), `false` | Whether the player's controls are visible at all times. With the default value (`true`), the controls are not displayed unless the user is hovering the mouse over the player. When this occurs, the controls remain visible while the user interacts with the player and for an additional period of time equal to the `controlBarAutoHideTimeout` setting. With a value of false, the controls are continuously visible and may reduce the amount of the player window available to display content. |
| Duration of control bar visibility | `controlBarAutoHide Timeout` | *Number* | A duration, in seconds, for the control bar to remain visible, after the user has stopped interacting with the player. Default is 3. This value is used when `controlBarAutoHide` is set to `true`. |

| Setting | Name | Possible values | Description |
|---|---|---|---|
| Pre-play poster frame | `poster` | *URL* | A URL specifying an image to display in the player window before playback begins. If no default poster image is provided, the player displays the current background color. The image must be in a bitmap image format, such PNG, BMP, or JPG. Vector images, movies, or animations are not supported. |
| Post-play poster frame | `endOfVideoOverlay` | *URL* | A URL specifying an image to display in the player window after playback completes. This value is set to the value of the `poster` setting by default, so that the same poster image appears at the end as at the beginning of playback. The same file type restrictions as for `poster` apply. |
| Play button overlay | `playButtonOverlay` | `true` (default), `false` | The default value displays a large Play button over the center of the player window before playback begins. |
| Looping behavior | `loop` | `false` (default), `true` | Restarts media playback when the end of the file is reached. The default behavior for the player is not to loop. |
| Display buffering indicator | `bufferingOverlay` | `true` (default), `false` | The default value displays a visual notification when playback is paused to refill the buffer. |
| Automatic playback | `autoPlay` | `false` (default), `true` | Starts playing the media automatically, without user input. The default behavior for the player is to require the user to start playback. |
| Mute the volume | `muted` | `false` (default), `true` | Specifies whether the player initially loads content with its volume on or off. |
| Set the volume | `volume` | *Number* | The initial volume of the media. Allowable values range from 0 (silent) to 1 (full volume). |
| Set the sound balance | `audioPan` | *Number* | The left-right sound volume balance for the media. Allowable values range from -1 (full pan left) to 1 (full pan right). A value of 0 sets both sides to an equal volume. |

# Advanced configuration options

As with the basic configuration settings, the advanced configuration options are the same for both Flash Media Playback and Strobe Media Playback.

In addition to some general advanced options, there are additional settings that control how the player manages bandwidth to improve the user's viewing experience.

For Flash Media Playback, you change these settings with the setup assistant. For Strobe Media Playback, you can set these options as described in "Configuring Strobe Media Playback" on page 10. You aren't required to adjust these settings for either player. Both players' default options are designed to perform well for most users.

## General advanced options

Ranging from display settings to debugging tools, both players provide a variety of advanced options you may configure. For a description of other advanced settings, see "Bandwidth management options" on page 8.

| Setting | Name | Possible values | Description |
|---|---|---|---|
| Media stream type | `streamType` | `liveOrRecorded` (default), `live`, `recorded`, `dvr` | The type of media stream to support. The default setting plays both live and recorded media, with no digital video recording (DVR) features. |
| XML configuration file | `configuration` | *URL* | A URL specifying the location of an XML configuration file. ***Note:*** *Support for external XML configuration files is under development and may change in the future.* |
| Player appearance/ "chrome" | `skin` | *URL* | A URL specifying the location of a file containing alternate images for the player's user interface elements. For more details, see Changing the Appearance of the Player |
| Method of scaling content | `scaleMode` | `letterbox` (default), `none`, `stretch`, `zoom` | Determines how the source content is sized within the player window. The default `letterbox` value allows the content to be resized to fit the player window, but constrains the dimensions of the content to maintain its original aspect ratio. A value of `none` does not allow the content to be resized. A value of `stretch` sets the dimensions of the content to that of the player window, possibly changing the aspect ratio of the content in the process. A value of `zoom` displays the content filling the player window, while maintaining its original aspect ratio; this may cause cropping of the content's horizontal or vertical edges. |
| Background color | `backgroundColor` | *Hexadecimal* | The color, specified as a hexadecimal value, to use for the background of the player. The player background is visible when no content is being played. The default color is black. |
| Produce full error messages | `verbose` | `false` (default), `true` | Whether to display detailed error messages for debugging. The default value (`false`) causes the display of simplified, user-friendly error messages. |
| Stop the player from loading if there is a plug-in error | `haltOnError` | `false` (default), `true` | If set to `true`, the player stops loading and displays an error if there is a problem loading a plug-in. |
| Sets the start time for a subclip | `clipStartTime` | *Number* | Specifies an offset in seconds from the beginning of the content stream. When specified, the stream is presented as a subclip, with playback beginning at the given start time. The default value is `NaN`, which starts playback at the actual beginning of the stream. Progressive content is unaffected by this setting. |

| Setting | Name | Possible values | Description |
|---|---|---|---|
| Sets the end time for a subclip | `clipEndTime` | *Number* | Specifies an offset in seconds from the beginning of the content stream. When specified, the stream is presented as a subclip, with playback stopping at the given end time. The default value is `NaN`, which plays the content to the actual end of the stream. Progressive content is unaffected by this setting. |
| Vertical pixel limit for standard quality video | `highQualityThreshold` | *Number* | The maximum vertical pixel resolution for which the video is treated as being of standard quality. With resolutions greater than this value, videos are considered to be high quality and the HD indicator is displayed in the "on" state. The default is 480, so videos with vertical resolutions of 720 pixels or 1080 pixels are considered to be high definition (HD) by default. The player uses this value to enable full-screen best practices, including disabling smoothing/deblocking filters for HD content. For standard definition content, the player enables smoothing/deblocking. |
| Specifies parsing of RTMP streaming URLs | `urlIncludesFMS ApplicationInstance` | `false` (default), `true` | When using Flash Media Server, you can connect to either an application or an application instance. Either way, you specify a stream name (and optionally one or more directories in which the stream is nested) as part of the connection path, as well. This can create a parsing issue, because it can be unclear what the part of the path between the application name and the stream name is identifying. For example, if you pass rtmp://www.myhost.com/myapp/**foo**/mystream, it is unknown whether "foo" is an application instance or a directory in which the stream is nested. To clarify this, the urlIncludesFMSApplicationInstance setting allows you to explicitly declare for RTMP streaming URLS whether the path contains an application instance or not. If `true`, then the URL does include an application instance, and foo must be a specific instance of myapp, and mystream must be the stream. If `false`, there is no instance of myapp in the URL, and foo must be a directory in which mystream is nested. The default is `false`. |

## Bandwidth management options

Both players provide features to help users obtain seamless, high-quality viewing experiences, whatever their bandwidth. Each aims to start playback as quickly as possible, while still creating a viewing experience with few, if any, pauses in playback. The features that support this are dynamic streaming and optimized buffering.

With **dynamic streaming**, when the player senses a network bandwidth change, it responds by switching playback to a content file with a more appropriate bitrate. Playback can switch among content files depending on the current bandwidth. This avoids having to pause playback entirely.

Dynamic streaming requires you to have multiple bitrate versions of content for the player to switch among, but the benefits to the user are significant. For a network experiencing a temporary reduction in bandwidth, playback does not have to pause for the user. Instead, the player seamlessly shifts to using a lower bitrate version of the content that is playing.

If it is enabled, dynamic streaming searches for multiple bitrate versions of the content to play. If it does not find multi-bitrate (MBR) versions, dynamic streaming does not function.

When **optimized buffering** is enabled, the player allows fast-start buffering for high-speed networks. For low-speed networks (where the network bandwidth is lower than the video bitrate), the player dynamically computes a buffer size that is sufficient to support continuous playback.

There are some situations where optimized buffering does not function.

- Dynamic streaming/MBR. With multi-bitrate content, optimized buffering is disabled to avoid interfering with the switching mechanism.
- Live/DVR content. With live or DVR-enabled streams, insufficient information is available to perform optimizations.

The following table describes the bandwidth management options for both players.

| Setting | Name | Possible values | Description |
|---|---|---|---|
| Remember viewer's bandwidth (allows dynamic streaming) | `optimizeInitialIndex` | `true` (default), `false` | The default value allows the player to use dynamic streaming for multi-bitrate (MBR) content. When the user starts playback, the player uses the download speed of the network connection to select the optimal starting bitrate stream. |
| Allow optimized buffering | `optimizeBuffering` | `true` (default), `false` | The default value allows fast-start buffering for high-speed networks and dynamic calculation of buffer size for low-speed networks. This setting has no effect on dynamic (MBR) streams or live/DVR content. |
| Length of buffer to create before starting playback | `initialBufferTime` | *Number* | The amount of the buffer (in seconds) that must be filled before playback begins. The default value is 0.1 second. If `optimizeBuffering` is set to `true`, the player uses this value when enabling fast-start buffering for high-speed networks. |
| Maximum allowed buffer length | `expandedBufferTime` | *Number* | The maximum size of the buffer (in seconds) that the player attempts to fill, once playback has begun. The default value is 10 seconds. If `optimizeBuffering` is set to `true`, the player uses this value along with fast-start buffering to optimize buffering for high-speed networks. |
| Minimum continuous playback duration | `minContinuousPlayback` | *Number* | The minimum amount of playback time without pausing to refill the buffer. The default value is 30 seconds. The player's optimized buffering algorithm uses this value to compute a target buffer size for low-speed networks. This setting only applies to streaming content, not progressive downloads. |

# Configuring Strobe Media Playback

There are three methods to set configuration options for Strobe Media Playback:

- Use the Flash Media Playback setup page to generate sample HTML code that you then customize. In this sample code, you must take care to replace the default location of the Flash Media Playback player with the location of your Strobe Media Playback player. To do so, replace the `object` tag's `movie` value and the `embed` tag's `src` value with the fully qualified path for the location of your player.
- Write HTML code yourself that specifies the settings for your configuration, as described in "Configuring Strobe Media Playback with FlashVars" on page 10.
- Provide a URL in your web page's code that gives the location of an XML file containing configuration settings. ***Note:*** *Strobe Media Playback support for external XML configuration files is under development and may change at a future time.*

However you set your options, they are applied dynamically. That is, your configuration choices are provided to the player at the time that it loads, not before.

## Configuring Strobe Media Playback with FlashVars

"Running Strobe Media Playback" on page 3 describes how to use FlashVars to specify the location of the content that you want to play. You can also use FlashVars to pass along configuration information to the player. When you do so, you instruct the player to ignore its default settings and use your customized choices.

***Note:*** *Configuration FlashVars in a page's HTML code are the final specifications for player options. They override any settings from an external XML configuration file, if one exists.*

Following the exact syntax for FlashVars is required. First, FlashVars must be placed within HTML `object` and `embed` tags. Second, the format of the FlashVars must be a set of one or more *"name=value"* pairs. Third, an ampersand (`&`) delimits each pair; for example, *"name1=value1&name2=value2"*.

The FlashVars marked in bold in the code below provide an example of the proper syntax. The tabs and bolding are used in the code for readability only.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=10,0,0
,0"
width="470" height="320">
    <param  name="movie"
            value="http://my.website.com/strobe/StrobeMediaPlayback.swf"> </param>
    <param  name="flashvars"
            value="src=http://my.website.com/strobe/stufftoplay/content.f4m
                &streamType=recorded
                &loop=true
                &autoPlay=true
                &playButtonOverlay=false
                &controlBarAutoHide=false"> </param>
    <param  name="allowFullScreen" value="true"></param>
    <param  name="allowscriptaccess" value="always"></param>
    <embed
        src="http://my.website.com/strobe/StrobeMediaPlayback.swf"
        type="application/x-shockwave-flash"
        allowscriptaccess="always" allowfullscreen="true" width="470" height="320"
        flashvars="src=http://my.website.com/strobe/stufftoplay/content.f4m
            &streamType=recorded
            &loop=true
            &autoPlay=true
            &playButtonOverlay=false
            &controlBarAutoHide=false">
    </embed>
</object>
```

# Chapter 3: Changing the Appearance of the Player

You can easily customize the look or "chrome" of your player's interface. This process is also known as providing a new skin for the player. Here are the basic steps:

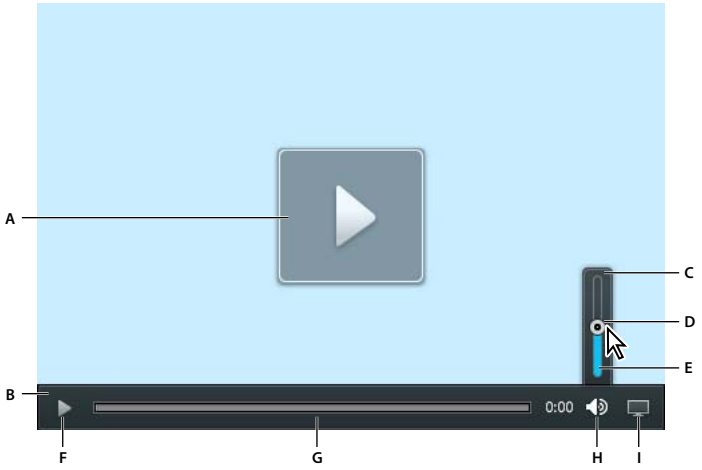
1. See "Identifying player interface elements" on page 12 for descriptions of each of the customizable pieces of the player's interface and a list of their IDs. Notice that many items are associate with multiple names. These typically refer to the different states (such as active or disabled) in which the element can be displayed. For visual consistency, provide a new look for each state of an element.
2. Create a custom bitmap image for each part of the interface that you want to change (for example, the Pause button). If you don't provide a custom image for a piece, it appears with its default skin. Save the bitmap image as a JPEG, GIF, PNG, or SWF file.
3. Be aware of the following limitations:
   a. While there are no restrictions on the sizes of your custom elements, the default spacing between items is not adjustable. Therefore, if you create images that are much larger or smaller than the default size, the interface can become confusing.
   b. There is currently no support for changing the default font, its size, or color.
   c. While JPEG, GIF, PNG, and SWF are all supported file types, using large animated SWF files can cause performance penalties if they are not in the browser cache.
   d. For security reasons, the player is not allowed to access data that resides outside the exact web domain from which it originates. However, to obtain custom skin files or to use a skin configuration XML file, the player may need to access a server other than the one on which it resides. If this is the case, those external servers must have cross-domain policy files (crossdomain.xml files) that give access permission to the player's server.
4. In the code for your web page, specify the interface element to replace and give the location of the image that replaces it. See "Replacing interface elements" on page 19 for examples and a description of this process.

## Identifying player interface elements

The following sections describe each of the customizable pieces of the player's interface and give their IDs.

Note that you are not required to customize all the interface elements, nor are all elements displayed at all times. Many parts of the player interface are only used with certain categories of content such as playlists or HD movies.

The following image provides an example of the player's interface elements while in a simple configuration and using a default skin.




*Simple default player skin elements*
***A.** Play button overlay **B.** Control bar background **C.** Volume bar background **D.** Volume bar slider **E.** Volume bar track **F.** Play button **G.** Scrub bar track **H.** Volume button **I.** Full screen button*

## Control bar backdrop and scrub bar control

The player's control bar is the horizontal strip that contains the player's controls. The control bar is by default located in a docked position at the bottom of the player window.

The scrub bar is a long, horizontal track; it is the largest single element on the control bar. The scrub bar is used to indicate the current state of media loading and playback.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| controlBarBackdrop | 2 x 35 | The background for the player's control bar. The image for this element is stretched as necessary to fill the length of the control bar's background area. |
| controlBarBackdropLeft | 2 x 35 | The left edge of the control bar background. This element creates a vertical border to the left side of the background. |
| controlBarBackdropRight | 2 x 35 | The right edge of the control bar background. This element creates a vertical border to the right side of the background. |
| scrubBarTrack | 2 x 9 | This image is used for the area of the scrub bar that represents unloaded content. The image for this element is stretched as necessary to fill the length of the track area. |
| scrubBarTrackLeft | 2 x 9 | The left edge of the scrub bar. This element creates a vertical border to the left side of the track. |
| scrubBarTrackRight | 2 x 9 | The right edge of the scrub bar. This element creates a vertical border to the right side of the track. |
| scrubBarLoadedTrack | 2 x 9 | The area of the scrub bar that represents loaded content. The image for this element is stretched as necessary to fill the length of the track's loaded-content area. |
| scrubBarLoadedTrackEnd | 2 x 9 | The edge of a scrub bar containing loaded content. This element creates a vertical border to the area of the track that represents loaded content. |

| Element ID | Default size (pixels) | Description |
|---|---|---|
| scrubBarPlayedTrack | 2 x 9 | The area of the scrub bar that represents played content. The image for this element is stretched as necessary to fill the length of the track's played-content area. |
| scrubBarDVRLiveTrack | 61 x 5 | The track representing DVR-enabled content is drawn within the border of the standard scrub bar track. This image is used when DVR-enabled live content is being played. |
| scrubBarDVRLiveInactiveTrack | 61 x 5 | This image is used when DVR-enabled live content is paused. |
| scrubBarLiveOnlyTrack | 61 x 5 | The track representing live content is drawn within the border of the standard scrub bar track. This image is used when live content that is not DVR-enabled is being played. |
| scrubBarLiveOnlyInactiveTrack | 61 x 5 | This image is used when live content that is not DVR-enabled is paused. |
| scrubBarScrubberNormal | 9 x 9 | The scrub bar control (also known as the "playhead" or "current-time indicator"). This element both indicates the current relative position of playback and provides an interface for the user to move playback backward or forward in time. In the "normal" state, the scrub bar control is active, but the user is not interacting with it. |
| scrubBarScrubberDown | 9 x 9 | The "down" state of the scrub bar control indicates that the user is currently selecting or dragging it. |
| scrubBarScrubberOver | 9 x 9 | The "over" state of the scrub bar control indicates that the user's pointer is hovering over the scrub bar control area. |
| scrubBarScrubberDisabled | 9 x 9 | The "disabled" state of the scrub bar control indicates that this feature is currently not available to the user. |

## Play/Pause button

The Play button and the Pause button are used together but do *not* appear concurrently. When playback is paused or has not begun, the Play button is displayed. Likewise, when playback is occurring, the Pause button is displayed. The Play and Pause buttons are displayed to the left of the scrub bar control.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| playButtonNormal | 24 x 24 | This image is used when the user has the option of starting playback, that is, when the content is currently paused. |
| playButtonDown | 24 x 24 | This image is used when the user has selected the Play button, but has not released it. |
| playButtonOver | 24 x 24 | This image is used when the user is moving the cursor over the Play button, but has not selected it. |
| pauseButtonNormal | 24 x 24 | This image is used when the user has the option of pausing playback. |
| pauseButtonDown | 24 x 24 | This image is used when the user has selected the Pause button, but has not released it. |

| Element ID | Default size (pixels) | Description |
|---|---|---|
| pauseButtonOver | 24 x 24 | This image is used when the user is moving the cursor over the Pause button, but has not selected it. |

## Sound control

The sound control consists of two main parts: the Volume button and the volume bar.

The Volume button is located on the right side of the control bar. In addition to being displayed in "normal," "down," and "over" states, the button is also displayed differently depending upon whether the audio is set to a low, medium, or high level. When the user selects the Volume button, the player mutes the sound, and the Unmute button is displayed in place of the Volume button.

Displayed vertically above the Volume button is the volume bar. Its elements consist of a background, a track to indicate volume level, and a "thumb" or slider to control the volume level.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| volumeButtonNormal | 19 x 24 | The Volume button in its active state, displayed without any indications of audio level. |
| volumeButtonDown | 19 x 24 | The Volume button in its selected state, displayed without any indications of audio level. |
| volumeButtonOver | 19 x 24 | The Volume button when the user is moving the cursor over it, displayed without any indications of audio level. |
| volumeButtonLowNormal | 19 x 24 | The Volume button in its active state, displaying an indication of a low audio level. |
| volumeButtonLowDown | 19 x 24 | The Volume button in its selected state, displaying an indication of a low audio level. |
| volumeButtonLowOver | 19 x 24 | The Volume button when the user is moving the cursor over it, displaying an indication of a low audio level. |
| volumeButtonMedNormal | 19 x 24 | The Volume button in its active state, displaying an indication of a medium audio level. |
| volumeButtonMedDown | 19 x 24 | The Volume button in its selected state, displaying an indication of a medium audio level. |
| volumeButtonMedOver | 19 x 24 | The Volume button when the user is moving the cursor over it, displaying an indication of a medium audio level. |
| volumeButtonHighNormal | 19 x 24 | The Volume button in its active state, displaying an indication of a high audio level. |
| volumeButtonHighDown | 19 x 24 | The Volume button in its selected state, displaying an indication of a high audio level. |
| volumeButtonHighOver | 19 x 24 | The Volume button when the user is moving the cursor over it, displaying an indication of a high audio level. |
| unmuteButtonNormal | 19 x 24 | The Volume button in its active state, indicating the volume is currently muted. |
| unmuteButtonDown | 19 x 24 | The Volume button in its selected state, indicating the volume is currently muted. |

| Element ID | Default size (pixels) | Description |
|---|---|---|
| unmuteButtonOver | 19 x 24 | The Volume button when the user is moving the cursor over it, indicating the volume is currently muted. |
| volumeBarBackdrop | 31 x 97 | The vertical background for the pop-up portion of the sound control. This background is always positioned above the Volume button. |
| volumeBarTrack | 5 x 2 | The vertical track that appears when the user selects the Volume button. This track is always displayed within the volumeBarBackdrop element. The image for this element is stretched as necessary to fill the length of the track. |
| volumeBarTrackEnd | 5 x 2 | The horizontal edge of the volume bar track. |
| volumeBarSliderNormal | 13 x 13 | The "thumb" or slider that the user moves to adjust the audio level. This value refers to the slider in its active state. |
| volumeBarSliderDown | 13 x 13 | The "down" state indicates that the user has selected the slider, but has not released it. |
| volumeBarSliderOver | 13 x 13 | The "over" state indicates that the user is moving the cursor over the slider, but has not selected it. |

## Next and Previous buttons

The Next and Previous buttons are used with playlists, where the player has multiple items to consecutively play. These buttons are displayed concurrently and are located between the Play/Pause button and the left edge of the scrub bar track.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| previousButtonNormal | 24 x 24 | The "normal" state indicates that there is a playlist item before that currently playing, and the user can press this button to select it. |
| previousButtonDown | 24 x 24 | The "down" state indicates that the user has selected this button, but has not released it. |
| previousButtonOver | 24 x 24 | The "over" state indicates that the user is moving the cursor over the button, but has not selected it. |
| previousButtonDisabled | 24 x 24 | This element is displayed when there is no prior track to play. |
| nextButtonNormal | 24 x 24 | The "normal" state indicates that there is a playlist item after that currently playing, and the user can press this button to select it. |
| nextButtonDown | 24 x 24 | The "down" state indicates that the user has selected this button, but has not released it. |
| nextButtonOver | 24 x 24 | The "over" state indicates that the user is moving the cursor over the button, but has not selected it. |
| nextButtonDisabled | 24 x 24 | This element is displayed when there is no track following the current one. |

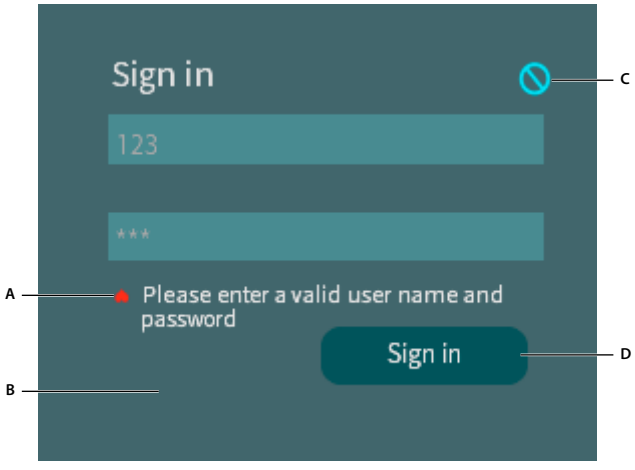
## Full Screen button and HD indicator

The Full Screen button allows the user to select whether to display the player window at a size covering their full computer screen. The HD indicator is not a user-selectable control; it indicates only whether high-definition (HD) content is currently playing. Both the Full Screen and HD elements are displayed on the right side of the player control bar.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| fullScreenEnterButtonNormal | 20 x 24 | The control the user selects to display the player window at a size that covers their full computer screen. This control is typically only displayed when the window is at less than full-screen size. The "normal" state indicates the control is active and selectable. |
| fullScreenEnterButtonDown | 20 x 24 | The "down" state indicates the user has selected the Full Screen control, but has not released it. |
| fullScreenEnterButtonOver | 20 x 24 | The "over" state indicates the user is moving the cursor over the Full Screen control, but has not selected it. |
| fullScreenLeaveButtonNormal | 20 x 24 | The control the user selects to display the player window at a size less than their full computer screen. This control is typically only displayed when the window is already at full-screen size. The "normal" state indicates the control is active and selectable. |
| fullScreenLeaveButtonDown | 20 x 24 | The "down" state indicates the user has selected the control, but has not released it. |
| fullScreenLeaveButtonOver | 20 x 24 | The "over" state indicates the user is moving the cursor over the control, but has not selected it. |
| hdOn | 21 x 24 | This element indicates that high-definition content is currently playing. |
| hdOff | 21 x 24 | This element indicates that high-definition content is not currently playing. |

## Authorization dialog

Both players display an authorization dialog when a user attempts to play content you have protected with Flash Access 2.0.




*Authorization dialog displayed when playing protected content*
***A.** Warning icon **B.** Authorization background **C.** Cancel button **D.** Submit button*

| Element ID | Default size (pixels) | Description |
|---|---|---|
| authWarning | 11 x 9 | The warning icon in the authorization dialog for protected content. This image is displayed when the user supplies an invalid name or password. |
| authBackdrop | 294 x 209 | The background of the authorization dialog for protected content. |
| authSubmitButtonNormal | 104 x 31 | The Submit button in the authorization dialog for protected content. In the "normal" state, the button is active and available for the user to select. |
| authSubmitButtonDown | 104 x 31 | The Submit button in the authorization dialog for protected content. In the "down" state, the user has selected this button, but has not released it. |
| authSubmitButtonOver | 104 x 31 | The Submit button in the authorization dialog for protected content. In the "over" state, the user is moving the cursor over the button, but has not selected it. |
| authCancelButtonNormal | 12 x 12 | The Cancel button in the authorization dialog for protected content. The "normal" state indicates that the button is active and available for the user to select. |
| authCancelButtonDown | 12 x 12 | The Cancel button in the authorization dialog for protected content. The "down" state indicates that the user has selected this button, but has not released it. |
| authCancelButtonOver | 12 x 12 | The Cancel button in the authorization dialog for protected content. The "over" state indicates that the user is moving the cursor over the button, but has not selected it. |

## Overlays

Both players use the following overlay images to provide additional information to the user.

| Element ID | Default size (pixels) | Description |
|---|---|---|
| scrubBarTimeHint | 65 x 43 | This element displays the current playback time. It is displayed as floating over the scrub bar, located at a position 35 pixels over the bottom of the control bar. |
| playButtonOverlayNormal | 116 x 107 | The player has an optional setting that displays a large Play button overlaying the middle of the screen, prior to the start of playback. The "normal" state for this button indicates that the user may start playback. |
| playButtonOverlayDown | 116 x 107 | The "down" state indicates that the user has selected the Play button overlay, but has not released it. |
| playButtonOverlayOver | 116 x 107 | The "over" state indicates that the user is moving the cursor over the Play button overlay, but has not selected it. |
| bufferingOverlay | 124 x 54 | This value specifies an image to use to indicate that the player is paused while filling its buffer. |

# Replacing interface elements

The final step in applying a custom skin is to tell the player what elements should be replaced and where the replacement files are located.

The simplest way to do this is to set a FlashVar `skin` variable in the page's HTML code. The `skin` variable must contain an `element` tag for each part of the skin you are changing. For more information on using FlashVars, see "Configuring Strobe Media Playback with FlashVars" on page 10. Alternately, you may provide the same skin information within an XML configuration file.

Each `element` tag must have an `id` attribute and a `src` attribute. The `id` attribute identifies the bitmap from the default skin to replace. The `src` attribute specifies the location (either relative to the player SWF file or an absolute path) of the custom bitmap to use. "Identifying player interface elements" on page 12 provides a comprehensive list of IDs for the player's skin elements.

Unless many of the replacement interface elements share the same base path, you typically use fully qualified URLs ("absolute" references) for each:

```
<skin>
    <element id="hdOn" src="http://www.myserver.com/myImages/hdOn.png"/>
    <element id="hdOff" src="http://www.myserver.com/myImages/hdOff.png"/>
    <element id="timeHint" src="http://www.webserver.com/sharedImages/timeHint.png"/
</skin>
```

However, if many elements do share the same base path, you can use partial URLs ("relative" references). To do this, use an `elements` tag to group the elements that share a base path. For the `elements` group, the shared path is specified by a `basePath` attribute. When set, all the `element` tags within the group get the `basePath` value prefixed to their `src` attribute:

```
<skin>
    <elements basePath="http://www.myserver.com/images/">
        <element id="hdOn" src="hdOn.png"/>
        <element id="hdOff" src="hdOff.png"/>
        <element id="timeHint" src="timeHint.png"/
    </elements>
</skin>
```

If you have several elements that do share a base path and several that do not, you can combine these methods. You can create an `elements` group and use relative paths for the ones that do share a base path. For the ones that do not, you must leave them outside the `elements` tag and specify their locations with absolute paths.

Note that the value for `basePath` is prefixed to the `element` tag values *without* adding any additional characters. For example, if `basePath` is `www.myserver.com/images` and an `elementsrc` value is `timeHint.png`, the resulting path is read as `www.myserver.com/imagestimeHint.png`, which is unlikely to be what you want.

Therefore, when using `basePath` for a folder, you must provide a URL that ends in a trailing "/". For example, `www.myserver.com/images/` would combine with `timeHint.png` to provide a result that accurately reflects the filename and directory structure.

# Chapter 4: Advanced Topics

A modern media player does much more than play media. It may also use a content delivery network (CDN), present advertising, capture user events to report to an analytics server, and so on. But the media player does not usually handle this work by itself. This additional functionality is typically provided in conjunction with third-party software known as "plug-ins." To make use of these external tools, see "Using plug-ins" on page 21.

The ability to multicast content is new with Flash Media Playback 1.5 and Strobe Media Playback 1.5. See "Delivery protocols and multicast content" on page 20 for more details.

## Delivery protocols and multicast content

The Flash Media Playback and Strobe Media Playback players automatically support a variety of delivery protocols:

- HTTP, for live and recorded streaming, as well as progressive download.
- RTMP, for live and recorded streams. (Note that, by convention, RTMP streams should *not* include a file extension as part of their URL.)

Additionally, new with Flash® Media Server 4.0, both players also support:

- RTMFP, to enable multicasting of live streams.

### About RTMFP multicast

Today's broadband audiences have created an urgent business need to support extreme-traffic situations, without pouring money into fixed server and bandwidth solutions. Even the most powerful Content Delivery Network (CDN) would be hard-pressed to keep up with the traffic that the London Olympics is expected to generate, for example.

Multicast is a way of leveraging existing bandwidth to distribute your content. With multicast, you are not limited to the bandwidth of your server(s). Instead, you can make use of intermediate distribution points, and even your viewers, to move your content along.

Extreme situations are not the only places where multicast is valuable, though. More commonly, within the enterprise, internal website managers and IT professionals can use multicast for dramatic savings on video delivery costs.

Using the Flash Media Playback or Strobe Media Playback players with RTMFP multicast, you can:

- stream continuous live video within an enterprise, such as a company meeting
- broadcast video within and beyond your network without a content delivery network (CDN)
- allow internal network clients to participate in a peer-to-peer (P2P) group, to help ensure high-quality video delivery

***Note:*** *Dynamic streaming and DVR functionality (such as pausing or stopping a video) are not supported with RTMFP multicast at this time.*

There are three forms of RTMFP multicast that the Flash Media Playback and Strobe Media Playback players support:

**IP multicast.** No matter the size of the receiving group for a given transmission, with IP multicast the sender needs to transmit only a single data packet. It is the intermediary IP routers in the network that are responsible for multiplying the transmission and sending the copies to receivers.

**P2P (application) multicast.** P2P multicast uses Flash Player applications to route and relay data, providing one-to-many (or a-few-to-many) streaming.

**Fusion multicast.** Fusion multicast combines IP multicast and P2P/application multicast. Clients can receive content via IP multicast or fall back to P2P/application multicast if IP multicast is not available.

***Note:*** *RTMFP multicast is a managed connection that requires the authorization of a Flash Media Server 4.0 instance to make the introductions. Clients must remain connected to the server to retain the direct connection.*

## Playing live content with multicast

Both the Flash Media Playback and Strobe Media Playback players automatically handle multicast content that is specified by a Flash Media Manifest (F4M) file. You can use the Flash Media Server "Configurator" tool to generate a multicast-enabled F4M manifest file.

***Note:*** *If you use the Flash Media Server 4.0.0 Configurator to generate the F4M file, you must manually change the names of two generated values. Change the name* `rtmfpGroupspec` *to* `groupspec`*, and the name* `rtmfpStreamName` *to* `multicastStreamName`*. If you use the FMS 4.0.1 Configurator, these names are already updated.*

To observe or debug the operation of multicast, you can right-click the control bar to activate its contextual menu. Then, select "Strobe Media Playback Info". This displays an overlay with a variety of current transmission statistics, including relevant values for multicast, buffering, and more. For further information on the Info Overlay, see the Strobe Media Playback wiki.

# Using plug-ins

A plug-in is nothing more than code that you invite to work with your player. When you load a plug-in, you give it permission to provide additional functionality for your player. Plug-ins are not given unlimited access to your media player. Flash Media Playback and Strobe Media Playback use Open Source Media Framework as a broker between your media player and the plug-in. This approach ensures that communication between media player and plug-in is both secure and standardized, making it simple to add, update, or switch plug-ins.

Your job is to load the plug-in by inserting a small amount of code in the HTML source for your web page. Just as you use FlashVars to customize your player's features or appearance, you also use them to load plug-ins. Specifically, you provide the location and any metadata for the plug-in within the FlashVars parameter.

You need some basic information from the plug-in's developer to begin:

1. A URL giving the location of the plug-in.
2. Whether a namespace or any other metadata is required for the plug-in to run. Note that plug-in metadata is specified in properties preceded by the plug-in's name and the underline character "_".

The bolded parts of this sample show plug-in specific values. These values include the location of the plug-in to load and metadata to provide the plug-in.

```
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=10,0,0
,0"
width="470" height="320">
    <param name="movie"
        value="http://www.myserver.com/strobe/StrobeMediaPlayback.swf"></param>
    <param name="flashvars" value="src=http://www.myserver.com/movie.flv&
        plugin_myPlugin1=http://www.mysite.com/plugins/myPlugin1.swf&
        myPlugin1_namespace=http://www.mysite.com/namespace/1.0&
        myPlugin1_retryLive=true&myPlugin1_retryInterval=10"></param>
    <param name="allowFullScreen" value="true"></param>
    <param name="allowscriptaccess" value="always"></param>
    <embed src="http://www.myserver.com/strobe/StrobeMediaPlayback.swf"
        type="application/x-shockwave-flash"
        allowscriptaccess="always" allowfullscreen="true"
        width="470" height="320"
        flashvars="src=http://www.myserver.com/mymovie.flv&
            plugin_myPlugin1=http://www.mysite.com/plugins/myPlugin1.swf&
            myPlugin1_namespace=http://www.mysite.com/namespace/1.0&
            myPlugin1_retryLive=true&myPlugin1_retryInterval=10">
    </embed>
</object>
```

For more information on OSMF plug-ins, including sample plug-ins, a development guide, and links to plug-ins developed by third parties, see www.osmf.org and opensource.adobe.com. For more information on using plug-ins with your Flash Media Playback or Strobe Media Playback player, see the Strobe Media Playback wiki.

## Plug-in whitelists

Flash Media Playback 1.5 and Strobe Media Playback 1.5 each provide an extra level of security by supporting a "whitelist" mechanism. A whitelist ensures that only plug-ins from specified hosts can access your player.

For Flash Media Playback, the whitelist operates automatically and is managed and updated by Adobe.

For Strobe Media Playback, the whitelist mechanism is *not* enabled by default. Whether to create a whitelist for your player is up to you. If you want to create a whitelist for your player, you can customize the sample preloader (`src/samples/preloader/src/Preloader.as`) available in the Strobe Media Playback 1.5 zip package, then recompile your player to incorporate the whitelist. All domain names on the list must be specified as fully qualified URLs, as only exact matches are given access.