VideoPlayer class

Using the VideoPlayer class

The VideoPlayer class lets you create a video player with a slightly smaller SWF file size than if you used the FLVPlayback component. Unlike the FLVPlayback component, the VideoPlayer class does not let you include a skin, and although you cannot find or seek to cue points, the `cuePoint` events will occur. The FLVPlayback class wraps the VideoPlayer class. Macromedia encourages you to use the FLVPlayback class in almost all cases, because there is no functionality in the VideoPlayer class that cannot be accessed using the FLVPlayback class.

Before you can use the VideoPlayer class, you must add the VideoPlayer movie clip instance from the FLVPlayback.fla external library and add it to your current document's library. For more information, see "Creating an application with the VideoPlayer class" on page 1.

Creating an application with the VideoPlayer class

The following procedure demonstrates how to add the VideoPlayer class to your Flash document by dragging it from an external library into your current document's library.

To create an application with the VideoPlayer class:

1. Create a new Flash document.
2. Select File > Import > Open External Library and navigate to:
   - On Windows: C:\Program Files\Macromedia\Flash 8*language*\Configuration\ComponentFLA\
   - On the Macintosh: Macintosh HD/Applications/Macromedia Flash 8/Configuration/ComponentFLA/
3. Select FLVPlayback.fla from the file browser and click Open.

4. Expand the FLVPlayback Assets folder in the FLVPlayback.fla Library panel and drag an instance of the VideoPlayer movie clip onto the Stage.

   Flash copies both the VideoPlayer movie clip and the VideoPlayerVideo Video object into the current document's library. You can now close the FLVPlayback.fla external library.

5. Select the VideoPlayer movie clip on the Stage and give it the instance name of **my_vp** using the Property inspector.

6. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

7. Select Control > Test Movie to test your Flash document.

   The Flash Video (FLV) file begins to play back in the video player instance on the Stage. Because the `VideoPlayer.autoSize` property is set to `true`, the `my_vp` instance on the Stage is automatically resized on the Stage to match the dimensions at which the FLV file was encoded.

To dynamically add a VideoPlayer class instance to your application:

1. Follow steps 1 through 4 of the previous procedure, "To create an application with the VideoPlayer class."

2. Delete the VideoPlayer movie clip instance from the Stage.

   When you delete the instance from the Stage, copies of both the VideoPlayer movie clip and the VideoPlayerVideo Video object remain in the current document's library.

3. Right-click (Windows) or Control-click (Macintosh) the VideoPlayer symbol in the Library panel, and select Linkage from the context menu.

4. Select the Export in First Frame option.

5. Click OK to close the dialog box.

6. Add the following ActionScript to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

/* Attach the VideoPlayer instance to the Stage and position
  the instance at (x: 100, y:100). */
this.attachMovie("VideoPlayer", "my_vp", 10, {x:100, y:100});
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

You attach the VideoPlayer instance to the Stage at runtime, and give it the instance name of **my_vp**. You then reposition the instance on the Stage by passing *x* and *y* values to the `attachMovie()` method.

7. Select Control > Test Movie to test the Flash document.

To apply filters to a VideoPlayer class instance:

1. Follow steps 1 through 5 of the earlier procedure, "To create an application with the VideoPlayer class."
2. Add the following ActionScript to Frame 1 of the main Timeline:

```
import flash.filters.ColorMatrixFilter;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var bw_array:Array = [0.3, 0.59, 0.11, 0, 0,
     0.3, 0.59, 0.11, 0, 0,
     0.3, 0.59, 0.11, 0, 0,
     0, 0, 0, 1, 0];
var myColorMatrix_filter:ColorMatrixFilter = new
  ColorMatrixFilter(bw_array);

var my_vp:VideoPlayer;
my_vp.autoSize = true;
my_vp.filters = [myColorMatrix_filter];
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");
```

3. Select Control > Test Movie to test the Flash document.

   The color matrix filter converts the contents of the video player to black and white.

NOTE	Because the VideoPlayer class inherits from the MovieClip class and doesn't implement the Macromedia Component Architecture, you can change parameters only by using ActionScript code and not by using the Parameters tab.

Streaming FLV files from a Flash Communication Server

If you use a Flash Communication Server to stream FLV files to the FLVPlayback component or VideoPlayer class instance, you must add the main.asc file to your Flash Communication Server FLV application. You can find the main.asc file in your Flash 8 application folder under Flash 8/Samples and Tutorials/Samples/Components/FLVPlayback/main.asc, or online at www.helpexamples.com/flash/videoplayer/main.asc.

To set up your Flash Communication Server for streaming FLV files:

1. Create a folder in your Flash Communication Server application folder, and give it a name such as **my_application**.
2. Copy the main.asc file into the my_application folder.
3. Create a folder named **streams** in the my_application folder.
4. Create a folder named **_definst_** inside the streams folder.
5. Place your FLV files in the **_definst_** folder.

   To access your FLV files on the Flash Communication Server, use a URL such as rtmp://*my_servername/my_application/stream.flv*.

   For more information on administering the Flash Communication Server, including how to set up a live stream, see the Flash Communication Server documentation at www.macromedia.com/support/documentation/en/flashcom/. When you play a live stream with Flash Communication Server, you need to use the `load()` or `play()` method to set the video player instance's `isLive` property to `true`. For more information, see `VideoPlayer.isLive` on page 38.

VideoPlayer class

Inheritance MovieClip > VideoPlayer class

ActionScript Class Name `mx.video.VideoPlayer`

The VideoPlayer class extends the MovieClip class and wraps a Video object. The FLVPlayback class wraps the VideoPlayer class. Macromedia encourages you to use the FLVPlayback class in almost all cases. You can access all functionalities in the VideoPlayer class with the FLVPlayback class.

The VideoPlayer class is documented because it lets you create a video player with a smaller SWF file. The VideoPlayer class does not allow you to include a skin, and it has a smaller API. You cannot find or seek to cue points, for example, although `cuePoint` events will occur.

In addition, the FLVPlayback class automatically interfaces with the NCManager class—for example, to access streaming FLV files on a Flash Communication Server. You interact with the NCManager class when you set the `contentPath` property and when you pass a URL to the `play()` and `load()` methods. If you use the VideoPlayer class by itself, however, you must include the following statement in your ActionScript code to make sure the NCManager class is included:

```
var _forceNCManager:mx.video.NCManager;
```

The NCManager class also has an interface class, INCManager, which lets you replace the NCManager class with a custom class for managing network communications. If you do that, include the following statement, replacing `NCManager` with the name of the class you have provided:

```
mx.video.VideoPlayer.DEFAULT_INCMANAGER = "mx.video.NCManager";
```

You do not need to add this statement if you use the default NCManager class.

NOTE	You can also set `DEFAULT_INCMANAGER` to replace the default `mx.video.NCManager` with the VideoPlayer instance.

Method summary for the VideoPlayer class

The following table lists the methods of the VideoPlayer class:

Method	Description
`VideoPlayer.addEventListener()`	Registers a listener for a specified event.
`VideoPlayer.close()`	Closes the video stream and the Flash Communication Server connection.
`VideoPlayer.load()`	Loads the FLV file but does not begin playing. After resizing (if needed), the FLV file is paused.
`VideoPlayer.pause()`	Pauses playing the video stream.
`VideoPlayer.play()`	Begins playing the video stream.
`VideoPlayer.removeEventListener()`	Removes an event listener.
`VideoPlayer.seek()`	Seeks to a specified time in the file, given in seconds, with decimal precision to milliseconds.
`VideoPlayer.setScale()`	Sets `scaleX` and `scaleY` simultaneously.
`VideoPlayer.setSize()`	Sets `width` and `height` simultaneously.
`VideoPlayer.stop()`	Stops playing the video stream.

Property summary for the VideoPlayer class

The VideoPlayer class has class and instance properties.

Class properties

The following properties occur only for the VideoPlayer class. They are constants that apply to all instances of the VideoPlayer class.

Property	Value	Description
`VideoPlayer.BUFFERING`	`"buffering"`	Read-only; possible value for the `state` property. Indicates the state entered immediately after `play()` or `load()` is called.
`VideoPlayer.CONNECTION_ERROR`	`"connectionError"`	Read-only; possible value for the `state` property. Indicates that a connection error occurred.

Property	Value	Description
VideoPlayer.DEFAULT_INCMANAGER	"mx.video.NCManager"	Name of the default (mx.video.NCManager) or custom implementation of the INCManager interface.
VideoPlayer.DISCONNECTED	"disconnected"	Read-only; possible value for the state property. Indicates that the FLV file stream is disconnected.
VideoPlayer.EXEC_QUEUED_CMD	"execQueuedCmd"	Read-only; state constant. Indicates the state during execution of the queued command.
VideoPlayer.LOADING	"loading"	Read-only; possible value for the state property. Indicates that the FLV file is loading.
VideoPlayer.PAUSED	"paused"	Read-only; possible value for the state property. Indicates that the FLV file is paused.
VideoPlayer.PLAYING	"playing"	Read-only; possible value for the state property. Indicates that the FLV file is playing.
VideoPlayer.RESIZING	"resizing"	Read-only; possible value for the state property. Indicates that the FLV file is resizing.
VideoPlayer.REWINDING	"rewinding"	Read-only; possible value for the state property. Indicates that the FLV file is rewinding.
VideoPlayer.SEEKING	"seeking"	Read-only; possible value for the state property. Indicates that the FLV file is seeking.
VideoPlayer.STOPPED	"stopped"	Read-only; possible value for the state property. Indicates that the FLV file is stopped.
VideoPlayer.version	x.x.x.xx	Read-only; the component's version number.

Instance properties

The following table lists the instance properties of the VideoPlayer class. This set of properties applies to each instance of a VideoPlayer class.

Property	Description
`VideoPlayer.autoRewind`	A Boolean value that, if `true`, causes the FLV file to rewind to the first frame when play stops.
`VideoPlayer.autoSize`	A Boolean value that, if `true`, causes the video to size automatically to the source dimensions.
`VideoPlayer.bufferTime`	A number that specifies the number of seconds to buffer in memory before beginning to play a video stream.
`VideoPlayer.bytesLoaded`	Read-only; a number that indicates the extent of downloading in number of bytes for an HTTP download.
`VideoPlayer.bytesTotal`	Read-only; a number that specifies the total number of bytes downloaded for an HTTP download.
`VideoPlayer.height`	A number that specifies the height of the video (in pixels).
`VideoPlayer.idleTimeout`	The amount of time, in milliseconds, before Flash terminates an idle connection to a Flash Communication Server because playing paused or stopped.
`VideoPlayer.isLive`	Read-only; a Boolean value that is `true` if the video stream is live. Not applicable to HTTP download.
`VideoPlayer.isRTMP`	Read-only; a Boolean value that is `true` if the FLV file is streaming from Flash Communication Server.
`VideoPlayer.maintainAspectRatio`	A Boolean value that, if `true`, maintains the video aspect ratio.
`VideoPlayer.metadata`	Read-only; an object that is a metadata information packet that is received from a call to the `onMetaData()` callback function, if available.
`VideoPlayer.ncMgr`	Read-only; an INCManager object that provides access to an instance of the class that implements INCManager.
`VideoPlayer.playheadTime`	A number that is the current playhead time or position, in seconds, which can be a fractional value.

Property	Description
`VideoPlayer.playheadUpdateInterval`	A number that is the amount of time, in milliseconds, between `playheadUpdate` events.
`VideoPlayer.progressInterval`	A number that is the amount of time, in milliseconds, between each `progress` event.
`VideoPlayer.scaleX`	A number that specifies the horizontal scale.
`VideoPlayer.scaleY`	A number that specifies the vertical scale.
`VideoPlayer.state`	Read-only; a string that specifies the state of the component. Set with the `load()`, `play()`, `stop()`, `pause()`, and `seek()` methods.
`VideoPlayer.stateResponsive`	Read-only; a Boolean value that is `true` if the state is responsive (that is, if controls can be enabled in the current state).
`VideoPlayer.totalTime`	Read-only; a number that is the total playing time for the video, in seconds.
`VideoPlayer.transform`	An object that provides direct access to the `Sound.setTransform()` and `Sound.getTransform()` methods to provide more sound control.
`VideoPlayer.url`	Read-only; a string that specifies the URL of the loaded (or loading) stream.
`VideoPlayer.videoHeight`	Read-only; a number that specifies the height of the FLV file.
`VideoPlayer.videoWidth`	Read-only; a number that specifies the width of the FLV file.
`VideoPlayer.visible`	A Boolean value that, if `true`, makes the FLV file visible.
`VideoPlayer.volume`	A number from 0 to 100 that indicates the volume control setting.
`VideoPlayer.width`	A number (percentage) that specifies how far a user can move the volume bar handle before an update occurs.
`VideoPlayer.x`	A number that specifies the horizontal dimension (in pixels) of the video player.
`VideoPlayer.y`	A number that specifies the vertical dimension (in pixels) of the video player.

Event summary for the VideoPlayer class

The following table lists the events of the VideoPlayer class:

Event	Description
`VideoPlayer.close`	Dispatched when the video stream is closed, whether through timeout or a call to the `close()` method.
`VideoPlayer.complete`	Dispatched when playing completes because the player reached the end of the FLV file.
`VideoPlayer.cuePoint`	Dispatched when a cue point is reached.
`VideoPlayer.metadataReceived`	Dispatched the first time the FLV file metadata is reached.
`VideoPlayer.playheadUpdate`	Dispatched every 0.25 seconds while the FLV file is playing.
`VideoPlayer.progress`	Dispatched every 0.25 seconds, starting when the `load()` method is called and ending when all bytes are loaded or a network error occurs.
`VideoPlayer.ready`	Dispatched when the FLV file is loaded and ready to display.
`VideoPlayer.resize`	Dispatched when the video is automatically resized.
`VideoPlayer.rewind`	Dispatched when the automatic rewind operation completes.
`VideoPlayer.stateChange`	Dispatched when the playback state changes.

VideoPlayer.addEventListener()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
my_videoPlayer.addEventListener(event:String, listener:Object)
```

Usage 2:

```
my_videoPlayer.addEventListener(event:String, listener:Function)
```

Parameters

- *event*:String The name of the event for which you are registering a listener. If the listener is an object, this is also the name of the listener object function to call.
- *listener*:Function or Object The name of the listener object or function that you are registering for the event.

Returns

Nothing.

Description

Method; registers a listener object or function for a specified event. If the listener is an object, the object must have a function defined for it with the same name as the event. If the listener is a function, the listener's name matches the function that is called to handle the event.

Example

The following example defines an event listener for the `stateChange` event that outputs the state and the current playhead time when certain states are entered at runtime.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doStateChange(eventObj:Object):Void {
  trace(eventObj.state + " (" + eventObj.playheadTime + " ms)");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading (0 ms)
playing (0.3 ms)
stopped (7.347 ms)
rewinding (7.347 ms)
stopped (0 ms)
```

See also

`VideoPlayer.removeEventListener()`

VideoPlayer.autoRewind

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.autoRewind

Description

Property; a Boolean value that, if `true`, causes the FLV file to rewind to Frame 1 when it stops playing, either because the player reached the end of the stream or the `stop()` method was called. This property is meaningless for live streams. The default value is `true`.

Example

The following example plays back a progressive download Flash video. When the playhead reaches the last video frame, the video is stopped and not reset to the first frame because the `autoRewind` property is set to `false`.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.autoRewind = false;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doStateChange(eventObj:Object):Void {
  trace(eventObj.state + " (" + eventObj.playheadTime + " ms)");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading (0 ms)
playing (0.3 ms)
stopped (7.347 ms)
```

VideoPlayer.autoSize

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.autoSize

Description

Property; a Boolean value that, if `true`, causes the video to size automatically to the dimensions of the source FLV file. If this property is set from `false` to `true` after an FLV file is loaded, the automatic resizing starts immediately. The default value is `false`.

Example

The following examples sets the `autoSize` property to `true`, which causes the VideoPlayer instance to resize itself to match the size of the Flash video.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

4. Select Control > Test Movie to test your Flash document.

The following example sets the `autoSize` property to `false` and resizes the VideoPlayer instance to 240 x 180 pixels. Because the `maintainAspectRatio` property is `true`, the VideoPlayer instance is then automatically resized to 240 x 161.25 pixels so the video object doesn't become distorted.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("ready", doReady)
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
```

```
my_vp.setSize(240, 180);
my_vp.play("http://www.helpexamples.com/flash/video/clouds.flv");
trace("before: {w:" + my_vp.width + ", h:" + my_vp.height + "}");

function doReady(eventObj:Object):Void {
  trace("after:  {w:" + my_vp.width + ", h:" + my_vp.height + "}");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
before: {w:240, h:180}
after:  {w:240, h:161.25}
```

VideoPlayer.BUFFERING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*BUFFERING

Description

Property (read-only); a state constant with the string value "buffering". You can compare this property to the state property to determine whether the component is in the buffering state. This state is entered immediately after play() or load() is called.

Example

The following example uses the stateChange event and a switch() statement to detect when the VideoPlayer object enters the BUFFERING state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.

4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.bufferTime = 6;
my_vp.load("http://www.helpexamples.com/flash/video/water.flv");

play_btn.onRelease = function() {
  my_vp.play();
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.BUFFERING:
    trace(eventObj.target + " is " + eventObj.state);
    break;
  }
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.state, VideoPlayer.stateResponsive

VideoPlayer.bufferTime

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.bufferTime

Description

Property; a number that specifies the number of seconds to buffer in memory before beginning to play a video stream. For FLV files streaming over RTMP, which are not downloaded and buffer only in memory, it can be important to increase this setting from the default value of 0.1 seconds. For a progressively downloaded FLV file over HTTP, little benefit is gained by increasing this value; however, increasing the value could improve viewing a high-quality video on an older, slower computer. The default value is 0.1 seconds.

NOTE	This property does not specify the amount of the FLV file to download before starting playback.

Example

The following example sets the buffer time for the progressively downloaded FLV file to 3 seconds.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.bufferTime = 3;
my_vp.autoRewind = true; // default
my_vp.autoSize = true;

// Change to your Flash Communication Server.
my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");
```

4. Select Control > Test Movie to test your Flash document.

VideoPlayer.bytesLoaded

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.bytesLoaded

Description

Property (read-only); a number that indicates the extent of downloading, in number of bytes, for an HTTP download. Returns -1 if no stream is present, if the stream is from a Flash Communication Server, or if the information is not yet available. The returned value is useful only for an HTTP download.

Example

The following example uses a ProgressBar component to display the loading progress of the Flash video.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("progress", doProgress);
my_vp.addEventListener("ready", doReady);
my_vp.autoRewind = true; // default
```

```
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";

play_btn.onRelease = function() {
  my_vp.play();
};

function doProgress(eventObj:Object):Void {
  trace("progress");
  my_pb.setProgress(eventObj.target.bytesLoaded,
  eventObj.target.bytesTotal);
}
function doReady(eventObj:Object):Void {
  trace("ready");
  my_pb.visible = false;
}
```

6. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.bytesTotal

VideoPlayer.bytesTotal

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.bytesTotal

Description

Property (read-only); a number that specifies the total number of bytes downloaded for an HTTP download. Returns -1 if no stream is present, if the stream is from a Flash Communication Server, or if the information is not yet available. The returned value is useful only for an HTTP download.

Example

The following example uses a ProgressBar component to display the loading progress of the Flash video.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("progress", doProgress);
my_vp.addEventListener("ready", doReady);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";

play_btn.onRelease = function() {
  my_vp.play();
};

function doProgress(eventObj:Object):Void {
  trace("progress");
  my_pb.setProgress(eventObj.target.bytesLoaded,
  eventObj.target.bytesTotal);
}
function doReady(eventObj:Object):Void {
  trace("ready");
  my_pb.visible = false;
}
```

6. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.bytesLoaded

VideoPlayer.close

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.close = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("close", listenerObject);
```

Usage 2:

```
on (close) {
  // ...
}
```

Description

Event; the VideoPlayer instance dispatches this event when it closes the NetConnection , by timing out or through a call to the `close()` method, when you call the `load()` method or the `play()` method, or set `contentPath` and cause the RTMP connection to close as a result. The VideoPlayer instance dispatches this event only when streaming from Flash Communication Server or Flash Video Streaming Service (FVSS). The event object has the properties `state` and `playheadTime`.

This event has the property `vp`, which is the index number of the video player to which this event applies.>

Property	Description
`state`	String; the state of the component (for example, `"disconnected"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example uses a button symbol instance to close a progressively downloaded video, and hide the video player on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the name **close_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("close", doClose);
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

close_btn.onRelease = function() {
  if (my_vp.state != VideoPlayer.DISCONNECTED) {
    my_vp.close();
  }
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  default :
    trace(eventObj.target + " is " + eventObj.state + " (" +
  eventObj.playheadTime + " ms)");
    break;
  }
}
function doClose(eventObj:Object):Void {
  trace("The video stream has been closed");
  eventObj.target.visible = false;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.close()

VideoPlayer.close()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.close()

Returns

Nothing.

Description

Method; forces the video stream and Flash Communication Server connection to close. This method triggers the `close` event. You usually do not need to call this method directly, because the idle timeout functionality takes care of closing the stream.

Example

The following example uses a button symbol instance to close a progressively downloaded video and hide the video player on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it a name of **close_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("close", doClose);
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

```
close_btn.onRelease = function() {
  my_vp.close();
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  default :
    trace(eventObj.target + " is " + eventObj.state + " (" +
  eventObj.playheadTime + " ms)");
    break;
  }
}
function doClose(eventObj:Object):Void {
  trace("The video stream has been closed");
  eventObj.target.visible = false;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.close, VideoPlayer.idleTimeout

VideoPlayer.complete

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.complete = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("complete", listenerObject);
```

Usage 2:

```
on (complete) {
  // ...
}
```

Description

Event; dispatched when playing completes because the player reached the end of the FLV file. The component does not dispatch the event if you call the `stop()` or `pause()` method, or click the corresponding controls. The event object has the properties `state` and `playheadTime`.

When the application uses progressive download, it does not set the `totalTime` property explicitly. If the video player downloads an FLV file that does not specify the duration in the metadata, the video player sets `totalTime` to an approximate total value before it dispatches this event.

The video player also dispatches the `stateChange` and `stopped` events.

Property	Description
`state`	String; the state of the component (for example, `"disconnected"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example traces a message to the Output panel after the video finishes playing.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("complete", doComplete);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doComplete(eventObj:Object):Void {
  trace(eventObj.target._name + " has completed");
}
```

4. Select Control > Test Movie to test your Flash document.

VideoPlayer.CONNECTION_ERROR

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*CONNECTION_ERROR

Description

Property (read-only); a state constant that contains the string value `"connectionError"`. You can compare this property to the `state` property to determine if a connection error state has occurred.

Example

The following example attempts to download a Flash video that isn't on the server, which causes a `CONNECTION_ERROR` state change.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;

// The following FLV file does not exist on the server and generates
// a connection error.

my_vp.play("http://www.helpexamples.com/flash/video/
  connection_error.flv");
```

```
function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.CONNECTION_ERROR :
    trace("Unable to load video");
    break;
  }
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.cuePoint

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.cuePoint = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("cuePoint", listenerObject);
```

Usage 2:

```
on (cuePoint) {
  // ...
}
```

Description

Event; dispatched when a cue point is reached. The event object has an `info` property that contains the `info` object received by the `NetStream.onCuePoint` callback for FLV file cue points. For ActionScript cue points, it contains the object that was passed into the ActionScript cue point methods or properties.

Property	Description
`info`	Object; a CuePoint object.

Example

The following example downloads an FLV file with embedded cue points. Each time the playhead reaches a cue point, the `cuePoint` event is dispatched and the cue point object appears in the Output panel.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("cuePoint", doCuePoint);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

function doCuePoint(eventObj:Object):Void {
  traceObject(eventObj.info, 0);
  trace("");
}

function traceObject(obj:Object, indent:Number):Void {
    var indentString:String = "";
    for (var j:Number = 0; j < indent; j++) {
        indentString += "\t";
    }
    for (var i:String in obj) {
        if (typeof(obj[i]) == "object") {
            trace(indentString + i + ": [Object]");
            traceObject(obj[i], indent + 1);
        } else {
            trace(indentString + i + ": " + obj[i]);
        }
    }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
parameters: [Object]
  lights: beginning
type: navigation
time: 0.418
name: point1

parameters: [Object]
  lights: middle
type: navigation
time: 7.748
name: point2

parameters: [Object]
  lights: end
type: navigation
time: 16.02
name: point3
```

VideoPlayer.DEFAULT_IDLE_TIMEOUT_INTERVAL

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

mx.video.VideoPlayer.DEFAULT_IDLE_TIMEOUT_INTERVAL

Description

Property; the default value for `VideoPlayer.idleTimeout`. The default value is 300,000 milliseconds (5 minutes).

Example

The following snippet traces the default idle timeout interval, which is set to 5 minutes (300,000 seconds):

```
trace(mx.video.VideoPlayer.DEFAULT_IDLE_TIMEOUT_INTERVAL); // 300000
```

See also

`VideoPlayer.idleTimeout`

VideoPlayer.DEFAULT_INCMANAGER

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*DEFAULT_INCMANAGER

Description

Property; name of the custom class, which all VideoPlayer objects that you create use as the default INCManager implementation. The default value is `"mx.video.NCManager"`.

Example

The following snippet traces the default INCManager , which is mx.video.NCManager:

```
trace(mx.video.VideoPlayer.DEFAULT_INCMANAGER); // mx.video.NCManager
```

The following example uses a custom progressive-download-only NCManager class.

1. Create a new Flash document and save it as **progressiveOnly.fla**.
2. Download the following ActionScript class and save it to the same folder as the file you created in step 1: www.helpexamples.com/flash/videoplayer/ProgressiveOnlyNCManager.as.
3. Switch to the Flash document you created in step 1.
4. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.VideoPlayer;

VideoPlayer.DEFAULT_INCMANAGER = "ProgressiveOnlyNCManager";

/* The following code forces the compiler to include the
   ProgressiveOnlyNCManager custom NCManager class in the
   current SWF file, which the VideoPlayer requires by default. */

var _forceINCManager:ProgressiveOnlyNCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

```
function doResize(eventObj:Object):Void {
  eventObj.target.x = (Stage.width - eventObj.width) / 2;
  eventObj.target.y = (Stage.width - eventObj.height) / 2;
}
```

6. Select Control > Test Movie to test your Flash document.

NOTE	ProgressiveOnlyNCManager is mostly a copy of the NCManager class, from which all code related to parsing SMIL and to handling streaming from Flash Communication Server has been removed. Using this class with VideoPlayer results in a slightly smaller SWF file than when you use the NCManager class (approximately 8.8 KB versus 13.1 KB with NCManager and SMILManager). These sizes reflect the classes themselves only and not additional ActionScript code needed to use the classes and other assets.

The following example uses a custom-streaming-only NCManager class:

1. Create a new Flash document and save it to your hard disk as **streamingOnly.fla**.
2. Download the following ActionScript class and save it to the same folder as the file you created in step 1: www.helpexamples.com/flash/videoplayer/StreamingOnlyNCManager.as.
3. Switch to the Flash document you created earlier.
4. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.VideoPlayer;

VideoPlayer.DEFAULT_INCMANAGER = "StreamingOnlyNCManager";

/* The following code forces the compiler to include the
  StreamingOnlyNCManager custom NCManager class in the
  current SWF file, which the VideoPlayer requires by default. */

var _forceINCManager:StreamingOnlyNCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = true;

// Change to your Flash Communication Server.

my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");

function doResize(eventObj:Object):Void {
  eventObj.target.x = (Stage.width - eventObj.width) / 2;
  eventObj.target.y = (Stage.width - eventObj.height) / 2;
}
```

6. Select Control > Test Movie to test your Flash document.

NOTE	StreamingOnlyNCManager is mostly a copy of NCManager class, from which all code related to parsing SMIL and to handling progressive download from Flash Communication Server has been removed. Using this class with the VideoPlayer class results in a slightly smaller SWF file than when you use the NCManager class (approximately 11.0 KB with the VideoPlayer class versus 13.1 KB with the NCManager and SMILManager classes). These sizes reflect the classes themselves and not additional ActionScript code that is needed to use the classes and other assets.

TIP	Although SMIL support is removed, you can still switch streams based on bit rate by using comma-delimited URLs—for example, rtmp://myfcs/mystream_low,50,mystream_dsl,128,mystream_high.

VideoPlayer.DEFAULT_UPDATE_PROGRESS_INTERVAL

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*DEFAULT_UPDATE_PROGRESS_INTERVAL

Description

Property; default value for `VideoPlayer.progressInterval`. The default value is 250 milliseconds (0.25 seconds).

Example

The following snippet traces the default update progress interval, which is set to one-fourth of a second:

```
trace(mx.video.VideoPlayer.DEFAULT_UPDATE_PROGRESS_INTERVAL); // 250
```

See also

`VideoPlayer.progressInterval`

VideoPlayer.DEFAULT_UPDATE_TIME_INTERVAL

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*DEFAULT_UPDATE_TIME_INTERVAL

Description

Property; default value for `VideoPlayer.playheadUpdateInterval`. The default value is 250 milliseconds (0.25 seconds).

Example

The following snippet traces the default update time interval, which is set to one-fourth of a second:

```
trace(mx.video.VideoPlayer.DEFAULT_UPDATE_TIME_INTERVAL); // 250
```

See also

`VideoPlayer.playheadUpdateInterval`

VideoPlayer.DISCONNECTED

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*DISCONNECTED

Description

Property (read-only); a state constant that contains the string value `"disconnected"`. You can compare this property to the `state` property to determine if a disconnected state exists.

Example

The following example plays a progressively downloaded FLV file. After the video finishes playing, the `complete` event is dispatched, which closes the video stream and hides the `my_vp` video player instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("complete", doComplete);
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doComplete(eventObj:Object):Void {
  eventObj.target.close();
}
function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.DISCONNECTED:
    trace(eventObj.target + " has been disconnected (" +
  eventObj.playheadTime + " ms)");
    eventObj.target.visible = false;
    break;
  default:
    trace(eventObj.state + " (" + eventObj.playheadTime + " ms)");
    break;
  }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading (0 ms)
playing (0.067 ms)
stopped (7.347 ms)
_level0.my_vp has been disconnected (7.347 ms)
```

See also

`VideoPlayer.close()`, `VideoPlayer.idleTimeout`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.EXEC_QUEUED_CMD

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``EXEC_QUEUED_CMD`

Description

Property (read-only); the state during the execution of the queued command.

NOTE	A `stateChange` event notification will not occur with this state; it is internal only.

See also

`VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.height

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.height`

Description

Property; a number that specifies the height of the VideoPlayer instance (in pixels). Not the same as `Video.height`, which is `videoHeight`.

Example

The following example uses the `autoSize` and `maintainAspectRatio` properties to automatically resize the video player instance to match the size of the progressively downloaded FLV file.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default

// default dimensions of the my_vp instance on the Stage

trace("before > w:" + my_vp.width + ", h:" + my_vp.height);

my_vp.load("http://www.helpexamples.com/flash/video/cuepoints.flv");

function doResize(eventObj:Object):Void {
  trace("after  > w:" + my_vp.width + ", h:" + my_vp.height);
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
before > w:160, h:120
after  > w:320, h:213
```

See also

`VideoPlayer.setSize()`, `VideoPlayer.videoHeight`, `VideoPlayer.width`

VideoPlayer.idleTimeout

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.idleTimeout

Description

Property; a number that specifies the amount of time, in milliseconds, before Flash terminates an idle connection to a Flash Communication Server because playing paused or stopped. This property has no effect on an FLV file downloading over HTTP.

If the `VideoPlayer.idleTimeout` property is set when a video stream is already idle, it restarts the timeout period with the new value.

The default value is 300,000 milliseconds (5 minutes).

Example

The following example assumes playing a streaming FLV file from a Flash Communication Server or FVSS. The example sets the `idleTimeout` property to a low value of 5000 milliseconds, which triggers a timeout and, consequently, a `close` event on the RTMP connection.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("close", doClose);
my_vp.autoSize = true;
my_vp.idleTimeout = 5000; // Close connection after 5 seconds.
```

```
// Change to your Flash Communication Server.

my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");

function doClose(eventObj:Object):Void {
  trace("Closed connection for video player: " + eventObj.target._name);
}
```

4. Select Control > Test Movie to test your Flash document.

   The RTMP connection is closed after approximately 5 seconds, because the Flash Communication Server connection is paused or stopped for longer than the specified `idleTimeout` value. The following text appears in the Output panel:

   ```
   Closed connection for video player: my_vp
   ```

See also

`VideoPlayer.DEFAULT_IDLE_TIMEOUT_INTERVAL`

VideoPlayer.isLive

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.isLive`

Description

Property (read-only); a Boolean value that is `true` if the video stream is live. This property is effective only when streaming from Flash Communication Server or Flash Video Streaming Service (FVSS). The value of this property is ignored for an HTTP download. To set this property, pass a value of `true` or `false` for the `isLive` parameter in the `load()` or `play()` method.

For more information, see "Streaming FLV files from a Flash Communication Server" on page 4.

Example

The following example loads a streaming video from a Flash Communication Server and uses the `isLive` property to display whether the stream is live.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("metadataReceived", doMetadataReceived);
my_vp.autoSize = true;

// Change to your Flash Communication Server.

my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");

function doMetadataReceived(eventObj:Object):Void {
  trace("isLive = " + eventObj.target.isLive); // false
  trace("isRTMP = " + eventObj.target.isRTMP); // true
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.load()`, `VideoPlayer.play()`

VideoPlayer.isRTMP

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.isRTMP

Description

Property (read-only); a Boolean value that is `true` if the FLV file is streaming from a Flash Communication Server or Flash Video Streaming Service (FVSS) using Real-Time Messaging Protocol (RTMP). Its value is `false` for any other FLV file source. The default value is `undefined`.

For more information, see "Streaming FLV files from a Flash Communication Server" on page 4.

Example

The following example loads a streaming video from a Flash Communication Server and uses the `isRTMP` property to display whether the stream is using the RTMP protocol.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer require
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("metadataReceived", doMetadataReceived);
my_vp.autoSize = true;

// Change to your Flash Communication Server.

my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");

function doMetadataReceived(eventObj:Object):Void {
  trace("isLive = " + eventObj.target.isLive); // false
  trace("isRTMP = " + eventObj.target.isRTMP); // true
}
```

4. Select Control > Test Movie to test your Flash document.

VideoPlayer.load()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

```
my_videoPlayer.load(url:String[, isLive:Boolean[, totalTime:Number]])
```

Parameters

- *url*:String A URL string for the FLV file that you want to load. If no value is passed for URL, an error is thrown with the message "null URL sent to VideoPlayer.load."
- *isLive:*Boolean [optional] The value is `true` if you stream a live feed from Flash Communication Server. The default value is `false`.
- *totalTime*:Number [optional] The length of an FLV file. Pass in 0, `null`, or `undefined` to automatically detect length from metadata, server, or XML. The default value is `undefined`.

Returns

Nothing.

Description

Method; similar to `play()`, but causes the FLV file to be loaded without playing. Autoresizing occurs if appropriate and the first frame of the FLV file is shown. After initial loading and autoresizing, the state is `PAUSED`.

This method takes the same parameters as the `play()` method, but you cannot call `load()` be called without a URL. If you do, an error is thrown. If the video player is in an unresponsive state, the `load()` method queues the request.

For more information, see "Streaming FLV files from a Flash Communication Server" on page 4.

Example

The following example loads a progressively downloaded FLV file and puts the video player instance into the `STOPPED` state. To begin the video playback, you must click the `play_btn` button on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/water.flv");

play_btn.onRelease = function() {
  my_vp.play();
};
```

5. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.isLive`, `VideoPlayer.play()`

VideoPlayer.LOADING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``LOADING`

Description

Property (read-only); a state constant that contains the string value `"loading"`. You can compare this property to the `state` property to determine whether the component is in the loading state.

Example

The following example uses the `stateChange` event and a `switch()` statement to detect when the VideoPlayer object enters the `LOADING` state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class
  requires by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("ready", doReady);
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/water.flv");

play_btn.onRelease = function() {
  my_vp.play();
};

function doReady(eventObj:Object):Void {
  trace(eventObj.target + " ready");
}
function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.LOADING:
    trace(eventObj.target + " loading");
    break;
  }
}
```

5. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.load()`, `VideoPlayer.play()`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.maintainAspectRatio

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.maintainAspectRatio

Description

Property; a Boolean value that, if `true`, maintains the video aspect ratio. If this property is changed from `false` to `true` and the `autoSize` property is `false` after an FLV file is loaded, an automatic resize of the video starts immediately. The default value is `true`.

Example

The following example uses the `maintainAspectRatio` property to ensure that the video isn't distorted after the video player instance is resized on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize)
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.setSize(240, 180);
my_vp.play("http://www.helpexamples.com/flash/video/clouds.flv");
trace("before: {w:" + my_vp.width + ", h:" + my_vp.height + "}");

function doResize(eventObj:Object):Void {
  trace("after:  {w:" + my_vp.width + ", h:" + my_vp.height + "}");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
before: {w:240, h:180}
after:  {w:240, h:161.25}
```

See also

`VideoPlayer.autoSize`

VideoPlayer.metadata

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.metadata

Description

Property (read-only); an object that is a metadata information packet that is received from a call to the `NetStream.onMetaData()` callback function, if available.

If the FLV file is encoded with the Flash 8 encoder, the `metadata` property contains the following information. Older FLV files contain only the `height`, `width`, and `duration` values. The default value is `null`.

Parameter	Description
`canSeekToEnd`	A Boolean value that is `true` if the FLV file is encoded with a keyframe on the last frame that allows seeking to the end of a progressive download movie clip. It is `false` if the FLV file is not encoded with a keyframe on the last frame.
`cuePoints`	An array of objects, one for each cue point embedded in the FLV file. The value is `undefined` if the FLV file does not contain any cue points. Each object has the following properties: • `type` A string that specifies the type of cue point as either `"navigation"` or `"event"`. • `name` A string that is the name of the cue point. • `time` A number that is the time of the cue point, in seconds, with a precision of three decimal places (milliseconds). • `parameters` An optional object that has name-value pairs that are designated by the user when creating the cue points.
`audiocodecid`	A number that indicates the audio codec (code-decode technique) that was used. Currently, the only value returned is 2, which represents the MP3 audio codec.
`audiodelay`	A number that indicates what time in the FLV file "time 0" of the original FLV file exists. The video content needs to be delayed for a brief time to properly synchronize the audio.
`audiodatarate`	A number that is the kilobytes per second of audio.
`videocodecid`	A number that is the codec version that was used to encode the video. Currently, the only values returned are 2 (which represents the Sorenson codec) or 4 (which represents the On2 codec).
`framerate`	A number that is the frame rate of the FLV file.
`videodatarate`	A number that is the video data rate of the FLV file.
`height`	A number that is the height of the FLV file.
`width`	A number that is the width of the FLV file.
`duration`	A number that specifies the duration of the FLV file, in seconds.

CAUTION

Due to the way objects are serialized within an FLV file, tracing a cue point object can cause several undefined values to be traced. To easily view the contents of a cue point, you can use a `for..in` loop, as demonstrated in the following example.

Example

The following example uses the `metadataReceived` event and a custom function, `traceObject()`, to recursively trace the contents of the FLV file's embedded metadata.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("metadataReceived", doMetadataReceived)
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.setSize(240, 180);
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

function doMetadataReceived(eventObj:Object):Void {
  traceObject(my_vp.metadata, 0);
}

function traceObject(obj:Object, indent:Number):Void {
    var indentString:String = "";
    for (var j:Number = 0; j < indent; j++) {
        indentString += "\t";
    }
    for (var i:String in obj) {
        if (typeof(obj[i]) == "object") {
            trace(indentString + i + ": [Object]");
            traceObject(obj[i], indent + 1);
        } else {
            trace(indentString + i + ": " + obj[i]);
        }
    }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
canSeekToEnd: true
cuePoints: [Object]
```

```
    2: [Object]
      parameters: [Object]
        lights: end
      type: navigation
      time: 16.02
      name: point3
    1: [Object]
      parameters: [Object]
        lights: middle
      type: navigation
      time: 7.748
      name: point2
    0: [Object]
      parameters: [Object]
        lights: beginning
      type: navigation
      time: 0.418
      name: point1
  audiocodecid: 2
  audiodelay: 0.038
  audiodatarate: 96
  videocodecid: 4
  framerate: 15
  videodatarate: 400
  height: 213
  width: 320
  duration: 16.334
```

See also

`VideoPlayer.load()`, `VideoPlayer.metadataReceived`, `VideoPlayer.play()`

VideoPlayer.metadataReceived

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.metadataReceived = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("metadataReceived", listenerObject);
```

Usage 2:

```
on (metadataReceived) {
  // ...
}
```

Description

Event; dispatched the first time the FLV file metadata is reached. The event object has an `info` property that contains the `info` object received by the `NetStream.onMetaData()` callback.

Property	Description
info	Object; the metadata object.

Example

The following example uses the `metadataReceived` event and a custom function, `traceObject()`, to recursively trace the contents of the FLV file's embedded metadata.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager class
  in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("metadataReceived", doMetadataReceived)
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.setSize(240, 180);
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

```
function doMetadataReceived(eventObj:Object):Void {
  traceObject(my_vp.metadata, 0);
}

function traceObject(obj:Object, indent:Number):Void {
    var indentString:String = "";
    for (var j:Number = 0; j < indent; j++) {
        indentString += "\t";
    }
    for (var i:String in obj) {
        if (typeof(obj[i]) == "object") {
            trace(indentString + i + ": [Object]");
            traceObject(obj[i], indent + 1);
        } else {
            trace(indentString + i + ": " + obj[i]);
        }
    }
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.metadata`

VideoPlayer.ncConnected()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer`.ncConnected()`

Returns

Nothing.

Description

Method; called by INCManager after the connection is complete or failed after a call to `INCManager.connectToURL`. If the connection failed, set `INCManager.nc` to `null` or `undefined` before calling.

Example

The following snippet is an excerpt from the custom INCManager implementation, StreamingOnlyNCManager.

```
private function tryFallBack():Void {
  if ((_serverName == fallbackServerName) ||
       (fallbackServerName == undefined) ||
       (fallbackServerName == null)) {
    // It's not connected.
    delete _nc;
    _nc = undefined;
    _owner.ncConnected();
  } else {
    cleanConns();
    _serverName = fallbackServerName;
    connectRTMP();
  }
}
```

To see a complete example using the StreamingOnlyNCManager class, see `VideoPlayer.DEFAULT_INCMANAGER`.

NOTE	To download the custom StreamingOnlyNCManager class, see www.helpexamples.com/flash/videoplayer/StreamingOnlyNCManager.as.

See also

`VideoPlayer.ncReconnected()`

VideoPlayer.ncMgr

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.ncMgr

Description

Property (read-only); an INCManager object that provides access to an instance of the class that implements INCManager, which is an interface to the NCManager class. You can use this property to implement a custom INCManager that requires custom initialization.

WARNING

Be careful when you loop over the `ncMgr` property and tracing values. If you write a function that recursively loops over the property, you may get recursion errors when working with RTMP streams, because a nested NetConnection object has pointers to the VideoPlayer instance that has a reference back to the original NetConnection object.

Example

The following example uses a `for..in` loop to trace each value within the `ncMgr` property for a video player instance on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager class
  in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("ready", doReady);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doReady(eventObj:Object):Void {
  var ncObj:Object = eventObj.target.ncMgr;
  for (var i:String in ncObj) {
    trace(i + ":\t" + ncObj[i]);
  }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
_nc:[object Object]
_owner:_level0.my_vp
_timeout:60000
_timeoutIntervalId:0
_tryNCIntervalId:0
_connTypeCounter:0
_payload:0
_autoSenseBW:false
_streams:undefined
_streamHeight:undefined
_streamWidth:undefined
_streamLength:undefined
_streamName:http://www.helpexamples.com/flash/video/sheep.flv
_contentPath:http://www.helpexamples.com/flash/video/sheep.flv
_appName:undefined
_portNumber:undefined
_wrappedURL:undefined
_serverName:undefined
_isRTMP:false
```

   If you connect to an RTMP stream, the following text appears in the Output panel:

```
_ncUri:rtmp://fcs.yourserver.com/rtmp_app_name
_nc:[object Object]
_protocol:rtmp:/
_owner:_level0.my_vp
_timeout:60000
_timeoutIntervalId:0
_tryNCIntervalId:0
_connTypeCounter:1
_payload:0
_autoSenseBW:false
_streams:undefined
_streamHeight:undefined
_streamWidth:undefined
_streamLength:2701.936
_streamName:rtmp_stream_name
_contentPath:rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name
_appName:broadcast
_portNumber:undefined
_wrappedURL:undefined
_serverName:fcs.yourserver.com
_isRTMP:true
```

The following example uses an NCManager instance to specify a fallback server in case the primary Flash Communication Server cannot be reached.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("rtmp://fcs-primary.yourserver.com/rtmp_app_name/
  rtmp_stream_name");

function doStateChange(eventObj:Object):Void {
  trace("state is: " + eventObj.state);
  if (eventObj.state == VideoPlayer.LOADING) {
    var my_ncm:NCManager = NCManager(eventObj.target.ncMgr);
    my_ncm.fallbackServerName = "fcs-secondary.yourserver.com";
  }
}
```

4. Select Control > Test Movie to test your Flash document.

   If the primary Flash Communication Server cannot be reached, the VideoPlayer instance attempts to connect to the secondary Flash Communication Server specified in the `fallbackServerName` property.

VideoPlayer.ncReconnected()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.ncReconnected()`

Returns

Nothing.

Description

Method; called by INCManager after the reconnection is complete or failed after a call to `INCManager.reconnect`. If the connection fails, set `INCManager.nc` to `null` before you call it.

Example

The following ActionScript code is an excerpt from the StreamingOnlyNCManager custom INCManager implementation:

```
public function reconnectOnStatus(target:NetConnection, info:Object):Void {
  if ((info.code == "NetConnection.Connect.Failed") ||
       (info.code == "NetConnection.Connect.Rejected")) {

    // Try the fallback server.

    delete _nc;
    _nc = undefined;
    _owner.ncReconnected();
  }
}
```

For a complete example that uses the StreamingOnlyNCManager class, see `VideoPlayer.DEFAULT_INCMANAGER`.

NOTE

To download the custom StreamingOnlyNCManager class, see **www.helpexamples.com/flash/videoplayer/StreamingOnlyNCManager.as**.

See also

`VideoPlayer.ncConnected()`

VideoPlayer.pause()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.pause()

Returns

Nothing.

Description

Method; pauses video playback. If video is paused or stopped, this method has no effect. To start playback again, call `play()`. If the video player is in an unresponsive state, the video player queues the request. This method throws an `mx.video.VideoError` exception if you call it when no stream is connected. You can use the `stateChange` event to determine when it is safe to call this method.

If the current state of the video player is `STOPPED`, the `pause()` method does nothing and the state of your video player remains stopped.

Example

The following example loads and plays a progressively downloaded FLV file. You can control video playback by clicking either the `play_btn` or `pause_btn` button symbol instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
4. Add a button symbol instance to the Stage and give it the instance name of **pause_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager class
  in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

```
play_btn.onRelease = function() {
  my_vp.play();
};
pause_btn.onRelease = function() {
  my_vp.pause();
};

function doStateChange(eventObj:Object):Void {
  trace(eventObj.type + ": " + eventObj.state + " (" +
  eventObj.playheadTime + " ms)");
}
```

6. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.play()`, `VideoPlayer.stateResponsive`

VideoPlayer.PAUSED

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer``.PAUSED`

Description

Property (read-only); a state constant that contains the string value `"paused"`. You can compare this property to the `state` property to determine if the component is in the paused state.

Example

The following example loads and plays a progressively downloaded Flash video. You can control video playback by clicking either the `play_btn` or the `pause_btn` button symbol instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.

4. Add a button symbol instance to the Stage and give it the instance name of **pause_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = false;
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.load("http://www.helpexamples.com/flash/video/water.flv");

play_btn.onRelease = function() {
  my_vp.play();
};

pause_btn.onRelease = function() {
  my_vp.pause();
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.PAUSED:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

6. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.load()`, `VideoPlayer.pause()`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.play()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

```
my_videoPlayer.play([url:String[, isLive:Boolean[, totalTime:Number]]])
```

Parameters

- *url*:String [optional] Pass in a URL string if you want to load and play a new FLV file. If you have already loaded an FLV file and want to continue playing it, pass in `null`.
- *isLive*:Boolean [optional] Pass in `true` if you are streaming a live feed from Flash Communication Server. The default value is `false`.
- *totalTime*:Number [optional] Pass in the length of the FLV file. Pass in 0, `null`, or `undefined` to automatically detect length from metadata, server, or XML. If `INCManager.streamLength` has a value other than 0, `null`, or `undefined` when `ncConnected()` is called, that value will trump the value of *totalTime*. . The default value is `undefined`.

Returns

Nothing.

Description

Method; causes the video to play. You can call this method while the video is paused, stopped, or playing. Call this method with no parameters to play a video that is already loaded or to pass in a URL to load a new stream. If the video player is in an unresponsive state, this method queues the request. The method throws an exception if called with no parameters and no stream is connected. Use the `stateChange` event to determine when it is safe to call this method.

For more information, see "Streaming FLV files from a Flash Communication Server" on page 4.

Example

The following example loads and plays a progressively downloaded FLV file. You can control video playback by clicking either the `play_btn` or `pause_btn` button symbol instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
4. Add a button symbol instance to the Stage and give it the instance name of **pause_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

play_btn.onRelease = function() {
  my_vp.play();
};
pause_btn.onRelease = function() {
  my_vp.pause();
};

function doStateChange(eventObj:Object):Void {
  trace(eventObj.type + ": " + eventObj.state + " (" +
  eventObj.playheadTime + " ms)");
}
```

6. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.isLive`, `VideoPlayer.load()`, `VideoPlayer.stateResponsive`

VideoPlayer.playheadTime

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.playheadTime

Description

Property; a number that specifies the playhead position (in seconds) since the start of the FLV file. Setting the `playheadTime` property directly does a seek and has all the restrictions of a seek.

The event `playheadUpdate` is dispatched when the playhead time changes, including every 0.25 seconds (or current value of `playheadUpdateInterval`) while the FLV file is playing.

Example

The following example loads a progressively downloaded FLV file and uses a ProgressBar component to display the playhead's current position.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.addEventListener("ready", doReady);
my_vp.maintainAspectRatio = true; // default
```

```
my_vp.autoSize = true;
my_vp.autoRewind = false;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";
my_pb.label = my_vp.state;
my_pb.indeterminate = true;

function doReady(eventObj:Object):Void {
  my_pb.label = eventObj.target.state;
  my_pb.indeterminate = false;
}
function doPlayheadUpdate(eventObj:Object):Void {
  my_pb.setProgress(eventObj.target.playheadTime,
  eventObj.target.totalTime);
  my_pb.label = eventObj.state;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.playheadTime, VideoPlayer.seek()

VideoPlayer.playheadUpdate

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.playheadUpdate = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("playheadUpdate", listenerObject);
```

Usage 2:

```
on (playheadUpdate) {
  //...
}
```

Description

Event; dispatched while the FLV file is playing at the frequency that the `playheadUpdateInterval` property specifies (the default is 0.25 seconds). The component does not dispatch this event when the video player is paused or stopped unless a seek occurs. The event object has the `state` and `playheadTime` properties.

Property	Description
`state`	String; the state of the component (for example, `"playing"`, `"stopped"`, or `"rewinding"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example loads a progressively downloaded FLV file and uses a ProgressBar component to display the playhead's current position. Whenever the `playheadUpdate` event is dispatched, the `ProgressBar.setProgress()` method is called and updates the progress bar.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.addEventListener("ready", doReady);
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.autoRewind = false;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

```
var my_pb:ProgressBar;
my_pb.mode = "manual";
my_pb.label = my_vp.state;
my_pb.indeterminate = true;

function doReady(eventObj:Object):Void {
  my_pb.label = eventObj.target.state;
  my_pb.indeterminate = false;
}
function doPlayheadUpdate(eventObj:Object):Void {
  my_pb.setProgress(eventObj.target.playheadTime,
  eventObj.target.totalTime);
  my_pb.label = eventObj.state;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.playheadUpdateInterval`

VideoPlayer.playheadUpdateInterval

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.playheadUpdateInterval`

Description

Property; a number that is the length of time, in milliseconds, between `playheadUpdate` events. Setting this property while the FLV file is playing restarts the timer. The default value is 250 milliseconds.

Because ActionScript cue points start on playhead updates, lowering the value of the `playheadUpdateInterval` property can increase the accuracy of ActionScript cue points. Because the playhead update interval is set by a call to the global `setInterval()` function, the update cannot occur more frequently than the SWF file frame rate, as with any interval that is set this way. So, as an example, for the default frame rate of 12 frames per second, the lowest effective interval that you can create is approximately 83 milliseconds, or 1 second (1000 milliseconds) divided by 12.

Example

The following example uses a ProgressBar component to display the percentage of data that is loaded into the video player instance. The `playheadUpdateInterval` property is set to 1000, which causes the `playheadUpdate` event to be dispatched once per second, which causes the progress bar instance to update.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.addEventListener("ready", doReady);
my_vp.autoSize = true;
my_vp.autoRewind = false;
my_vp.maintainAspectRatio = true; // default
// dispatch playheadUpdate event every 1000 milliseconds (1 second)
my_vp.playheadUpdateInterval = 1000;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";
my_pb.label = my_vp.state;
my_pb.indeterminate = true;

function doReady(eventObj:Object):Void {
  my_pb.label = eventObj.target.state;
  my_pb.indeterminate = false;
}
function doPlayheadUpdate(eventObj:Object):Void {
  my_pb.label = eventObj.state;
  my_pb.setProgress(eventObj.target.playheadTime,
  eventObj.target.totalTime);
}
```

5. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.playheadUpdate`

VideoPlayer.PLAYING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``PLAYING`

Description

Property (read-only); a state constant that contains the string value `"playing"`. You can compare this property to the `state` property to determine whether the component is in the `PLAYING` state.

Example

The following example uses the `stateChange` event and a `switch()` statement to detect when the VideoPlayer object enters the `PLAYING` state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = false;
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

```
function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.PLAYING:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.play()`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.progress

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.progress = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("progress", listenerObject);
```

Usage 2:

```
on (progress) {
  // ...
}
```

Description

Event; dispatched at the frequency that the `progressInterval` property specifies (the default is every 0.25 seconds), starting when the load begins and ending when all bytes are loaded or a network error occurs.

Dispatched only for a progressive HTTP download. Indicates progress in number of downloaded bytes. The event object has the `bytesLoaded` and `bytesTotal` properties, which are the same as the VideoPlayer properties of the same names.

Property	Description
`bytesLoaded`	Number; indicates the extent of downloading, in bytes, for an HTTP download.
`bytesTotal`	Number; specifies the total number of bytes downloaded for an HTTP download.

Example

The following example uses the progress event to update a ProgressBar instance, which displays which percentage of a progressively downloaded FLV file has downloaded.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("progress", doProgress);
my_vp.addEventListener("ready", doReady);
my_vp.bufferTime = 6;
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";
```

```
play_btn.onRelease = function() {
  my_vp.play();
};

function doProgress(eventObj:Object):Void {
  trace("progress");
  my_pb.setProgress(eventObj.target.bytesLoaded,
  eventObj.target.bytesTotal);
}
function doReady(eventObj:Object):Void {
  trace("ready");
  my_pb.visible = false;
}
```

6. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.progressInterval`

VideoPlayer.progressInterval

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.progressInterval

Description

Property; a number that specifies the length of time, in milliseconds, between each `progress` event. If you set this property while the video stream is playing, the timer restarts. The default value is 250 milliseconds (0.25 seconds).

Example

The following example sets the `progressInterval` property to 50 milliseconds, which causes the progress event to be dispatched 20 times per second.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("progress", doProgress);
my_vp.addEventListener("ready", doReady);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.progressInterval = 50;
my_vp.load("http://www.helpexamples.com/flash/video/lights_long.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";

play_btn.onRelease = function() {
  my_vp.play();
};

function doProgress(eventObj:Object):Void {
  trace("progress event at " + getTimer() + " ms");
  my_pb.setProgress(eventObj.target.bytesLoaded,
  eventObj.target.bytesTotal);
}
function doReady(eventObj:Object):Void {
  trace("ready");
  my_pb.visible = false;
}
```

6. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.progress

VideoPlayer.ready

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.ready = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("ready", listenerObject);
```

Usage 2:

```
on (ready) {
  // ...
}
```

Description

Event; dispatched when an FLV file is loaded and ready to display. It starts the first time you enter a responsive state after you load a new FLV file with the `play()` or `load()` method. It starts only once for each FLV file that is loaded.

The event object has the `state` and `playheadTime` properties.

Property	Description
`state`	String; the state of the component (for example, `"playing"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example uses the `ready` event to hide a progress bar component instance on the Stage after an FLV file is loaded and ready to display.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.

4. Add a button symbol instance to the Stage and give it the instance name of **play_btn**.
5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("progress", doProgress);
my_vp.addEventListener("ready", doReady);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.load("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";

play_btn.onRelease = function() {
  my_vp.play();
};

function doProgress(eventObj:Object):Void {
  trace("progress");
  my_pb.setProgress(eventObj.target.bytesLoaded,
  eventObj.target.bytesTotal);
}
function doReady(eventObj:Object):Void {
  trace("ready");
  my_pb.visible = false;
}
```

6. Select Control > Test Movie to test your Flash document.

VideoPlayer.removeEventListener()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
my_videoPlayer.removeEventListener(event:String, listener:Object)
```

Usage 2:

```
my_videoPlayer.removeEventListener(event:String, listener:Function)
```

Parameters

- *event*:String The name of the event for which you are removing a listener.
- *listener*:Function or Object A reference to the listener object or function that you are removing.

Returns

Nothing.

Description

Method; removes an event listener from a VideoPlayer instance.

Example

The following example uses the `playheadUpdate` event to remove an event listener after the FLV file's playhead reaches 2 seconds.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;
```

```
var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doStateChange(eventObj:Object):Void {
  trace(eventObj.type + ": " + eventObj.state);
}
function doPlayheadUpdate(eventObj:Object):Void {
  trace(eventObj.type + ": " + eventObj.state);
  if (eventObj.playheadTime > 2) {
    my_vp.removeEventListener("playheadUpdate", doPlayheadUpdate);
    trace("Removing event listener for \"" + eventObj.type + "\"
  event.");
  }
}
```

4. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.addEventListener()

VideoPlayer.resize

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.resize = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("resize", listenerObject);
```

Usage 2:

```
on (resize) {
  // ...
}
```

Description

Event; dispatched when the video is automatically resized. This occurs when either the `autoSize` or `maintainAspectRatio` property is `true`. This event is not dispatched when you call either the `setScale()` or `setSize()` method.

The event object has the `x`, `y`, `width`, and `height` properties.

Property	Description
`x`	Number; specifies the horizontal coordinate (location) of the video player.
`y`	Number; specifies the vertical coordinate (location) of the video player.
`width`	Number; specifies the width of the VideoPlayer instance on the Stage.
`height`	Number; specifies the height of the VideoPlayer instance on the Stage.

Example

The following example uses the `resize` event to trace the current values for the video player's *x* coordinate, *y* coordinate, width, and height.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.setSize(120, 120);
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object):Void {
  trace("x: " + eventObj.x);
  trace("y: " + eventObj.y);
  trace("width: " + eventObj.width); // 120
  trace("height: " + eventObj.height); // 90
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.autoSize`, `VideoPlayer.maintainAspectRatio`

VideoPlayer.RESIZING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``RESIZING`

Description

Property (read-only); a state constant that contains the string value `"resizing"`. You can compare this property to the `state` property to determine whether the component is in the `RESIZING` state.

Example

The following example uses the `resize` event to trace the current values for the video player's *x* coordinate, *y* coordinate, width, and height.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.addEventListener("stateChange", doStateChange);
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true; // default
```

```
// Change to your Flash Communication Server.

my_vp.load("rtmp://fcs.yourserver.com/rtmp_app_name/rtmp_stream_name");


function doResize(eventObj:Object):Void {
  trace("resize: " + eventObj.target.state);
}
function doStateChange(eventObj:Object):Void {
  if (eventObj.state == VideoPlayer.RESIZING) {
    trace("stateChange: VideoPlayer.RESIZING");
  } else {
    trace("stateChange: " + eventObj.state);
  }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
stateChange: loading
stateChange: VideoPlayer.RESIZING
resize: resizing
stateChange: stopped
```

See also

`VideoPlayer.autoSize`, `VideoPlayer.maintainAspectRatio`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.rewind

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.rewind = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("rewind", listenerObject);
```

Usage 2:

```
on (rewind) {
  // ...
}
```

Description

Event; dispatched when the location of the playhead moves backward when an automatic rewind completes.

The `stateChange` event is dispatched with a state of `rewinding` when an automatic rewind occurs. The `stateChange` event does not start until rewinding is complete. The `seek` event is dispatched when rewinding occurs through seeking. The VideoPlayer instance also dispatches the `playheadUpdate` event when rewinding occurs.

The event object has the properties `state` and `playheadTime`.

Property	Description
`state`	String; the state of the component (for example, `"stopped"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example traces the loaded the FLV file's URL once the video player dispatches the `rewind` event.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("rewind", doRewind);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doRewind(eventObj:Object):Void {
  trace("Rewinding " + eventObj.target.url);
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.autoRewind`

VideoPlayer.REWINDING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``REWINDING`

Description

Property (read-only); a state constant that contains the string value `"rewinding"`. You can compare this property to the `state` property to determine if the component is in the rewinding state. This state occurs only when the `autoRewind` property is set to `true`.

Example

The following example uses the `stateChange` event and a `switch()` statement to detect when the VideoPlayer object enters the `REWINDING` state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager class
  in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.REWINDING:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading [0 ms]
playing [0 ms]
stopped [7.347 ms]
> REWINDING [7.347 ms]
stopped [0 ms]
```

See also

`VideoPlayer.autoRewind`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.scaleX

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.scaleX

Description

Property; a number that is the horizontal scale. The default value is 100%.

Example

The following example uses the `scaleX` property to resize a video player instance on the Stage to 60% of its encoded width and height.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
// dimensions of VideoPlayer instance on Stage
trace("my_vp.width: " + my_vp.width + " px"); // 160 px
trace("my_vp.height: " + my_vp.height + " px"); // 120 px
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.scaleX = 60; // scale VideoPlayer instance to 60% size
trace("setting scale...")
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object):Void {
  // dimensions of encoded FLV video file
  trace("videoWidth: " + eventObj.target.videoWidth + " px"); // 320 px
```

```
    trace("videoHeight: " + eventObj.target.videoHeight + " px"); // 240
    px
    // dimensions of VideoPlayer instance on Stage (60% scale applied)
    trace("width: " + eventObj.target.width + " px"); // 96 px
    trace("height: " + eventObj.target.height + " px"); // 72 px
    // x-scale and y-scale of VideoPlayer instance on Stage
    trace("scaleX: " + eventObj.target.scaleX + "%"); // 60%
    trace("scaleY: " + eventObj.target.scaleY + "%"); // 60%
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.scaleY`, `VideoPlayer.setScale()`

VideoPlayer.scaleY

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.scaleY`

Description

Property; a number that is the vertical scale. The default value is 100%.

Example

The following example uses the `scaleY` property to resize a video player instance on the Stage to 60% of its encoded width and height.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
// dimensions of VideoPlayer instance on Stage
trace("my_vp.width: " + my_vp.width + " px"); // 160 px
trace("my_vp.height: " + my_vp.height + " px"); // 120 px
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.scaleY = 60; // scale VideoPlayer instance to 60% size
trace("setting scale...")
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object):Void {
  // dimensions of encoded FLV video file
  trace("videoWidth: " + eventObj.target.videoWidth + " px"); // 320 px
  trace("videoHeight: " + eventObj.target.videoHeight + " px"); // 240
  px
  // dimensions of VideoPlayer instance on Stage (60% scale applied)
  trace("width: " + eventObj.target.width + " px"); // 96 px
  trace("height: " + eventObj.target.height + " px"); // 72 px
  // x-scale and y-scale of VideoPlayer instance on Stage
  trace("scaleX: " + eventObj.target.scaleX + "%"); // 60%
  trace("scaleY: " + eventObj.target.scaleY + "%"); // 60%
}
```

4. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.scaleX, VideoPlayer.setScale()

VideoPlayer.seek()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.seek(*time:Number*)

Parameters

- *time*:Number The number of seconds to seek to. An `mx.video.VideoError` error is generated if the time isn't numeric (that is, `null` or `undefined`) or if the value is less than 0.

Returns

Nothing.

Description

Method; seeks to a given time in the file, specified in seconds, with a precision of three decimal places (milliseconds).

The `playheadTime` property might not have the expected value immediately after calling one of the seek methods or setting `playheadTime` to cause seeking. For a progressive download, you can seek only to a keyframe; therefore, a seek takes you to the time of the first keyframe after the specified time.

NOTE	When streaming, a seek always goes to the precise specified time even if the source FLV file doesn't have a keyframe there.

Seeking is asynchronous, so if you call a seek method or set the `playheadTime` property, `playheadTime` does not update immediately. To obtain the time after the seek is complete, listen for the `seek` event, which does not start until the `playheadTime` property is updated.

Example

The following example uses two button symbols, `beginning_btn` and `end_btn`, to seek to the beginning or end of the currently loaded FLV file.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add a button symbol instance to the Stage and give it the instance name of **beginning_btn**.
5. Add a button symbol instance to the Stage and give it the instance name of **end_btn**.
6. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.autoRewind = false;
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";

beginning_btn.onRelease = function() {
  my_vp.seek(0); // Seek to beginning of FLV.
  my_vp.play();
};
end_btn.onRelease = function() {
  my_vp.seek(my_vp.totalTime); // Seek to end of FLV.
};

function doPlayheadUpdate(eventObj:Object):Void {
  my_pb.setProgress(eventObj.target.playheadTime,
  eventObj.target.totalTime);
  my_pb.label = eventObj.state;
}
```

7. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.stateResponsive

VideoPlayer.SEEKING

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

*mx.video.VideoPlayer.*SEEKING

Description

Property (read-only); a state constant that contains the string value "seeking". You can compare this property to the state property to determine whether the component is in the seeking state.

Example

The following example uses the stateChange event and a switch() statement to detect when the VideoPlayer object enters the SEEKING state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **beginning_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoRewind = false;
my_vp.autoSize = true;
my_vp.maintainAspectRatio = true; // default
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

beginning_btn.onRelease = function() {
  my_vp.seek(0);
```

```
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.SEEKING:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

5. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

See also

`VideoPlayer.seek()`, `VideoPlayer.state`, `VideoPlayer.stateResponsive`

VideoPlayer.setScale()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.setScale(`*`scaleX:Number, scaleY:Number`*`)`

Parameters

- *`scaleX`*:Number A number representing the horizontal scale.
- *`scaleY`*:Number A number representing the vertical scale.

Returns

Nothing.

Description

Method; sets the `scaleX` and `scaleY` properties simultaneously. Setting these properties simultaneously can be more efficient than setting the `scaleX` and `scaleY` properties individually, because setting either one individually can cause automatic resizing.

If `autoSize` is `true`, this method has no effect, because the player sets its own dimensions. If the `maintainAspectRatio` property is `true` and `autoSize` is `false`, changing `scaleX` or `scaleY` causes automatic resizing.

Example

The following example uses the `setScale()` method to resize the video player instance on the Stage to 50% of its current size on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
// dimensions of VideoPlayer instance on Stage
trace("my_vp.width: " + my_vp.width + " px"); // 160 px
trace("my_vp.height: " + my_vp.height + " px"); // 120 px
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = false; // default
my_vp.maintainAspectRatio = true; // default
my_vp.setScale(50, 50);
trace("setting scale...")
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object):Void {
  // dimensions of encoded FLV video file
  trace("videoWidth: " + eventObj.target.videoWidth + " px"); // 320 px
  trace("videoHeight: " + eventObj.target.videoHeight + " px"); // 240
  px
  // dimensions of VideoPlayer instance on Stage (50% scale applied)
  trace("width: " + eventObj.target.width + " px"); // 80 px
  trace("height: " + eventObj.target.height + " px"); // 60 px
  // x-scale and y-scale of VideoPlayer instance on Stage
  trace("scaleX: " + eventObj.target.scaleX + "%"); // 50%
  trace("scaleY: " + eventObj.target.scaleY + "%"); // 50%
}
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.scaleX`, `VideoPlayer.scaleY`

VideoPlayer.setSize()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.setSize(*width:Number, height:Number*)

Parameters

- *width*:Number The width of the video player.
- *height*:Number The height of the video player.

Returns

Nothing.

Description

Method; sets the width and height simultaneously. Setting these properties simultaneously can be more efficient than setting the `width` and `height` properties individually, because setting either one individually can cause automatic resizing,

If `autoSize` is `true`, this method has no effect, because the player sets its own dimensions. If `maintainAspectRatio` is `true` and `autoSize` is `false`, changing the width or height causes automatic resizing.

Example

The following example uses the `setSize()` method to resize an instance on the Stage to 240 x 180 pixels. Because the `maintainAspectRatio` property is set to `true`, the video player instance is automatically resized to 240 x 161.25 pixels so the video does not become distorted.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize)
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.setSize(160, 180);
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");
trace("before: {w:" + my_vp.width + ", h:" + my_vp.height + "}");

function doResize(eventObj:Object):Void {
  trace("after:  {w:" + my_vp.width + ", h:" + my_vp.height + "}");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
before: {w:160, h:180}
after:  {w:160, h:120}
```

See also

`VideoPlayer.autoSize`, `VideoPlayer.height`, `VideoPlayer.maintainAspectRatio`, `VideoPlayer.width`

VideoPlayer.state

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.state`

Description

Property (read-only); a string that specifies the state of the component. This property is set by the `load()`, `play()`, `stop()`, `pause()`, and `seek()` methods. The possible values for the `state` property are `buffering`, `connectionError`, `disconnected`, `loading`, `paused`, `playing`, `rewinding`, `seeking`, and `stopped`.

Example

The following example uses the `state` property to toggle between the `PLAYING` and `STOPPED` states when you click the video player instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("ready", doReady);
my_vp.autoRewind = false;
my_vp.maintainAspectRatio = true; // default
my_vp.load("http://www.helpexamples.com/flash/video/water.flv");

// toggle between PLAYING and STOPPED states
my_vp.onRelease = function() {
  switch (my_vp.state) {
    case VideoPlayer.PLAYING:
      this.stop();
      break;
    case VideoPlayer.STOPPED:
      this.play();
      break;
  }
}

function doReady(eventObj:Object):Void {
  my_vp.autoSize = false; // default
  eventObj.target.width = 240;
}
```

4. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.BUFFERING, VideoPlayer.CONNECTION_ERROR, VideoPlayer.DISCONNECTED, VideoPlayer.load(), VideoPlayer.LOADING, VideoPlayer.pause(), VideoPlayer.PAUSED, VideoPlayer.play(), VideoPlayer.PLAYING, VideoPlayer.REWINDING, VideoPlayer.seek(), VideoPlayer.SEEKING, VideoPlayer.stop(), VideoPlayer.STOPPED

VideoPlayer.stateChange

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

Usage 1:

```
var listenerObject:Object = new Object();
listenerObject.stateChange = function(eventObject:Object):Void {
  // ...
};
my_videoPlayer.addEventListener("stateChange", listenerObject);
```

Usage 2:

```
on (stateChange) {
  // ...
}
```

Description

Event; dispatched when the video player's state changes at runtime. The event object has the properties `state` and `playheadTime`.

You can use this event to track when the video player enters or leaves unresponsive states during playback (such as in the middle of connecting, resizing, or rewinding), during which times the `play()`, `pause()`, `stop()`, and `seek()` methods queue the requests to be executed when the player enters a responsive state.

Property	Description
`state`	String; the state of the component (for example, `"loading"`, `"playing"`, `"stopped"`, `"disconnected"`, or `"rewinding"`).
`playheadTime`	Number; the current playhead time, in seconds.

Example

The following example uses the `stateChange` event to trace when the video player instance enters different states.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doStateChange(eventObj:Object):Void {
  trace(eventObj.target + " is " + eventObj.state + " (" +
  eventObj.playheadTime + " ms)");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
_level0.my_vp is loading (0 ms)
_level0.my_vp is playing (0.067 ms)
_level0.my_vp is stopped (7.347 ms)
_level0.my_vp is rewinding (7.347 ms)
_level0.my_vp is stopped (0 ms)
```

VideoPlayer.stateResponsive

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.stateResponsive

Description

Property (read-only); a Boolean value that is `true` if the state is responsive. If the state is unresponsive, calls to the `play()`, `load()`, `stop()`, `pause()` and `seek()` methods are queued by the video player and executed when the video player changes to a responsive state. Because these calls are queued and executed later, it is usually not necessary to track the value of the `stateResponsive` property. The responsive states are `disconnected`, `stopped`, `playing`, `paused`, and `buffering`.

Example

The following example uses the `stateChange` event to send information to the Output panel when the video player instance enters different states, and to indicate whether that state is responsive.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

function doStateChange(eventObj:Object):Void {
  trace("state: " + eventObj.state + " (stateResponsive: " +
  eventObj.target.stateResponsive + ")");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
state: loading (stateResponsive: false)
state: playing (stateResponsive: true)
state: stopped (stateResponsive: true)
state: rewinding (stateResponsive: false)
state: stopped (stateResponsive: true)
```

See also

VideoPlayer.CONNECTION_ERROR, VideoPlayer.DISCONNECTED, VideoPlayer.LOADING, VideoPlayer.PAUSED, VideoPlayer.PLAYING, VideoPlayer.RESIZING, VideoPlayer.REWINDING, VideoPlayer.STOPPED

VideoPlayer.stop()

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.stop()

Returns

Nothing.

Description

Method; stops video playback. If `autoRewind` is set to `true`, rewinds to the first frame. If the video is already stopped, the method has no effect. To start playback again, call `play()`. If the player is in an unresponsive state, the video player queues the request.

The method throws an exception if it's called when a stream is not connected. Use the `stateChange` event to determine when it is safe to call this method.

Example

The following example uses the `stateChange` event and a `switch` statement to detect when the video player instance is in a `STOPPED` state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button component instance to the Stage and give it the instance name of **stop_btn.**
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

stop_btn.onRelease = function() {
  my_vp.stop();
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.STOPPED:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

5. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading [0 ms]
playing [0.367 ms]
> STOPPED [7.347 ms]
rewinding [7.347 ms]
> STOPPED [0 ms]
```

See also

`VideoPlayer.autoRewind`, `VideoPlayer.play()`, `VideoPlayer.stateResponsive`

VideoPlayer.STOPPED

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer.``STOPPED`

Description

Property (read-only); a state constant with the string value `"stopped"`. The FLV file is loaded, and playback of the FLV file has stopped. This state is entered when the `stop()` method is called, and when the playhead reaches the end of the stream.

Example

The following example uses the `stateChange` event and a `switch()` statement to detect when the VideoPlayer object enters the `STOPPED` state.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button component instance to the Stage and give it the instance name of **stop_btn**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("stateChange", doStateChange);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");
```

```
stop_btn.onRelease = function() {
  my_vp.stop();
};

function doStateChange(eventObj:Object):Void {
  switch (eventObj.state) {
  case VideoPlayer.STOPPED:
    trace("> " + eventObj.state.toUpperCase() + " \t[" +
  eventObj.playheadTime + " ms]");
    break;
  default:
    trace(eventObj.state + " \t[" + eventObj.playheadTime + " ms]");
  }
}
```

5. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
loading [0 ms]
playing [0.197 ms]
> STOPPED [16.34 ms]
rewinding [16.34 ms]
> STOPPED [0 ms]
```

See also

`VideoPlayer.state`, `VideoPlayer.stateResponsive`, `VideoPlayer.stop()`

VideoPlayer.totalTime

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.totalTime`

Description

Property (read-only); a number that specifies the total running time of the FLV file in seconds. A value of 0, `null`, or `undefined` means that property was not passed into `play()` or `load()`, and was unable to detect automatically, or have not yet.

Example

The following example uses a ProgressBar component instance on the Stage to display the playback progress of the progressively downloaded video, as well as the current state of the video playback instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a ProgressBar component instance to the Stage and give it the instance name of **my_pb**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.ProgressBar;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("playheadUpdate", doPlayheadUpdate);
my_vp.addEventListener("ready", doReady);
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.autoRewind = false;
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

var my_pb:ProgressBar;
my_pb.mode = "manual";
my_pb.label = my_vp.state;
my_pb.indeterminate = true;

function doReady(eventObj:Object):Void {
  my_pb.label = eventObj.target.state;
  my_pb.indeterminate = false;
}
function doPlayheadUpdate(eventObj:Object):Void {
  my_pb.setProgress(eventObj.target.playheadTime,
  eventObj.target.totalTime);
  my_pb.label = eventObj.state;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.playheadTime

VideoPlayer.transform

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.transform

Description

Property; an object that provides direct access to the Sound.setTransform() and Sound.getTransform() methods to provide sound control. You must set this property to an object to initialize it and for changes to take effect. Reading the property provides you with a copy of the current settings, which you can change.

The default value is an object with the following values:

Property	Default value	Description
ll	100	A percentage value that specifies how much of the left input to play in the left speaker (0 to 100).
lr	0	A percentage value that specifies how much of the right input to play in the left speaker (0 to 100).
rl	0	A percentage value that specifies how much of the left input to play in the right speaker (0 to 100).
rr	100	A percentage value that specifies how much of the right input to play in the right speaker (0 to 100).

Example

The following example loops over each item in the sound transform object for the VideoPlayer object.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

var soundTransform_obj:Object = my_vp.transform;
for (var i:String in soundTransform_obj) {
  trace(i + ":\t" + soundTransform_obj[i]);
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
rl:0
rr:100
lr:0
ll:100
```

The following example sets a new sound transform object for the VideoPlayer object so that all of the audio comes from the left channel instead of both the left and right channels.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.transform = {ll:100, lr:100, rl:0, rr:0};
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.volume`

VideoPlayer.url

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.url

Description

Property (read-only); the URL of the currently loaded (or loading) stream. This URL is the URL that is last sent to `play()` or `load()`; `null` is sent if no stream is loaded.

Example

The following example displays the URL currently playing Flash Video file.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");
trace(my_vp.url); // http://www.helpexamples.com/flash/video/water.flv
```

4. Select Control > Test Movie to test your Flash document.

See also

`VideoPlayer.load()`, `VideoPlayer.play()`

VideoPlayer.version

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`mx.video.VideoPlayer``.version`

Description

Property (read-only); a class property that contains the VideoPlayer class's version number.

WARNING	Attempting to access the version property on an instance instead of the class causes the following compiler error: "Static members can only be accessed directly through classes."

Example

The following example sends the current version of the VideoPlayer class to the Output panel:

```
trace(mx.video.VideoPlayer.version);
```

VideoPlayer.videoHeight

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.videoHeight`

Description

Property (read-only); the source height of a loaded FLV file. This property returns `undefined` if no information is available yet.

Example

The following example uses the `resize` event to trace the dimensions of a progressively downloaded FLV file.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object) {
  trace(eventObj.target.url + " original dimensions are {vW:" +
  eventObj.target.videoWidth + ", vH:" + eventObj.target.videoHeight +
  "}");
  trace(eventObj.target._name + " is resizing to {w:" +
  eventObj.target.width + ", h:" + eventObj.target.height + "}");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
http://www.helpexamples.com/flash/video/sheep.flv original dimensions
  are {vW:320, vH:240}
my_vp is resizing to {w:160, h:120}
```

See also

`VideoPlayer.height`, `VideoPlayer.videoWidth`

VideoPlayer.videoWidth

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.videoWidth

Description

Property (read-only); the source width of the loaded FLV file. This property returns `undefined` if no information is available yet.

Example

The following example uses the `resize` event to trace the dimensions of a progressively downloaded FLV file.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = false; // default
my_vp.play("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object) {
  trace(eventObj.target.url + " original dimensions are {vW:" +
  eventObj.target.videoWidth + ", vH:" + eventObj.target.videoHeight +
  "}");
  trace(eventObj.target._name + " is resizing to {w:" +
  eventObj.target.width + ", h:" + eventObj.target.height + "}");
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
http://www.helpexamples.com/flash/video/sheep.flv original dimensions
  are {vW:320, vH:240}
my_vp is resizing to {w:160, h:120}
```

See also

`VideoPlayer.videoHeight`, `VideoPlayer.width`

VideoPlayer.visible

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

`my_videoPlayer``.visible`

Description

Property; a Boolean value that, if `true`, makes the VideoPlayer instance visible. If `false`, it makes the instance invisible. The default value is `true`.

Example

The following example uses two button symbol instances, `show_btn` and `hide_btn`, to control the video player instance's visibility.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a button symbol instance to the Stage and give it the instance name of **show_btn**.
4. Add a button symbol instance to the Stage and give it the instance name of **hide_btn**.

5. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

show_btn.onRelease = function() {
  my_vp.visible = true;
};
hide_btn.onRelease = function() {
  my_vp.visible = false;
};
```

6. Select Control > Test Movie to test your Flash document.

VideoPlayer.volume

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.volume

Description

Property; a number in the range of 0 to 100 that indicates the volume control setting. The default value is 100.

Example

The following example uses a NumericStepper component instance to control the volume level of a video player instance.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add a NumericStepper component instance to the Stage and give it the instance name of **volume_nstep**.
4. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.controls.NumericStepper;
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
   class in the current SWF file, which the VideoPlayer class requires
   by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.maintainAspectRatio = true; // default
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/cuepoints.flv");

var volume_nstep:NumericStepper;
volume_nstep.minimum = 0;
volume_nstep.maximum = 100;
volume_nstep.value = my_vp.volume;
volume_nstep.addEventListener("change", doChange);

function doChange(eventObj:Object) {
  my_vp.volume = eventObj.target.value;
}
```

5. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.transform

VideoPlayer.width

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.width

Description

Property; a number that specifies the width of the VideoPlayer instance on the Stage.

Example

The following example uses the `resize` event to trace the dimensions of the video player instance on the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoRewind = true; // default
my_vp.autoSize = true;

// default dimensions of video player object on Stage:
trace("before > w:" + my_vp.width + ", h:" + my_vp.height);

my_vp.load("http://www.helpexamples.com/flash/video/sheep.flv");

function doResize(eventObj:Object):Void {
  trace("after  > w:" + my_vp.width + ", h:" + my_vp.height);
}
```

4. Select Control > Test Movie to test your Flash document.

   The following text appears in the Output panel:

```
before > w:160, h:120
after  > w:320, h:240
```

See also

`VideoPlayer.height`, `VideoPlayer.setSize()`, `VideoPlayer.videoWidth`

VideoPlayer.x

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.x

Description

Property; a number that specifies the horizontal position (in pixels) of the video player.

Example

The following example uses the `resize` event to reposition a video player instance in the middle of the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.
3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;
```

```
var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doResize(eventObj:Object):Void {
  eventObj.target.x = Math.round((Stage.width - eventObj.width) / 2);
  eventObj.target.y = Math.round((Stage.height - eventObj.height) / 2);
}
```

4. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.y

VideoPlayer.y

Availability

Flash Player 8.

Edition

Flash Professional 8.

Usage

my_videoPlayer.y

Description

Property; a number that specifies the vertical position (in pixels) of the video player.

Example

The following example uses the `resize` event to reposition a video player instance in the middle of the Stage.

1. Create a new Flash document.
2. Add a VideoPlayer instance to the Stage (see "Creating an application with the VideoPlayer class" on page 1) and give it the instance name of **my_vp**.

3. Add the following ActionScript code to Frame 1 of the main Timeline:

```
import mx.video.NCManager;
import mx.video.VideoPlayer;

/* The following code forces the compiler to include the NCManager
  class in the current SWF file, which the VideoPlayer class requires
  by default. */

var _dummy:NCManager;

var my_vp:VideoPlayer;
my_vp.addEventListener("resize", doResize);
my_vp.autoSize = true;
my_vp.play("http://www.helpexamples.com/flash/video/water.flv");

function doResize(eventObj:Object):Void {
  eventObj.target.x = Math.round((Stage.width - eventObj.width) / 2);
  eventObj.target.y = Math.round((Stage.height - eventObj.height) / 2);
}
```

4. Select Control > Test Movie to test your Flash document.

See also

VideoPlayer.x