RealNetworks

EMBEDDED REALPLAYER EXTENDED FUNCTIONALITY GUIDE

Version 2

Revision Date: 25 September 2001

CONTENTS

INTRODUCTION

An *Embedded RealPlayer* is a variation of RealPlayer® that enables RealMedia™ content to be played back directly within a Web page, rather than launching RealPlayer as a separate application. Content played in an Embedded RealPlayer is called an *embedded presentation*.

You can create an Embedded RealPlayer for your Web page using the Netscape plug-in or ActiveX control that comes with the RealPlayer installation. By including the plug-in or control in your HTML application, you gain access to methods that set and retrieve presentation attributes, control the clip playback, and handle user interactions. While the methods are most commonly accessed from Java, JavaScript, or VBScript, they can also be developed using C++ and other programming languages.

Before you begin embedding presentations in your web page, you should first obtain and read the *RealSystem iQ Production Guide* available at **http://service.real.com/help/library/encoders.html**. This manual describes how to assemble a RealSystem presentation that can be used with the Embedded RealPlayer controls.

What's New in Embedded RealPlayer

Version 2 of this guide includes significant changes to the Netscape plug-in and ActiveX control API sets, improved sample applications, and a new callback event handling mechanism. The following sections highlight these changes. For more detailed descriptions of the changes, see the remaining chapters in this guide.

Methods Removed From the ActiveX and Netscape API Sets

Several methods have been removed from the ActiveX and Netscape APIs and are no longer supported. Calls to any of these functions will be ignored by the Embedded RealPlayer because no new methods have been introduced to the API sets to replace the removed methods.

For More Information: For a complete list of all removed methods, see "Methods Removed From the ActiveX and Netscape API Sets" on page 124.

Obsolete Methods in the ActiveX and Netscape API Sets

Several methods are considered to be obsolete in versions later than RealPlayer 8.x, but may be used depending on your development platform and application requirements. In some instances, new methods have been added to the API sets to replace some of the obsolete methods. When this is the case, it is strongly recommended that you use the new method.

For More Information: For a complete list of all obsolete methods, see "Obsolete Methods in the ActiveX and Netscape API Sets" on page 125.

New Event Handling Mechanism for Netscape Plug-ins

Netscape® version 6.0 does not support callback event handling in the same manner as previous versions. For this reason, Embedded RealPlayer introduces a new mechanism for event handling involving the use of JavaScript and callback methods in a Netscape plug-in. The procedure is available for Netscape running on Microsoft Windows® or Macintosh®, and Microsoft Internet Explorer® running on Macintosh. For more information about using the new mechanism, see "Handling Callback Events From RealPlayer" on page 12.

RealSystem Components

You need the following tools to create and test your embedded presentation:

- RealPlayer

Use RealPlayer, available free at **http://www.real.com**, to test your JavaScript or VBScript extensions. The RealPlayer installation includes the Netscape plug-in and ActiveX control for embedded playback.

- RealSystem Server

  RealSystem Server streams clips to RealPlayer. The server is not necessary for testing local playback of embedded clips, but is necessary for streaming presentations to RealPlayer over a network. The *RealSystem Server Administration Guide* is available at **http://service.real.com/help/library/servers.html**.

- RealSystem Software Development Kit (SDK)

  The RealSystem SDK is not necessary for extending RealPlayer's embedded playback features, but is required for advanced programming tasks such as building a new client interface on top of the RealPlayer core. A knowledge of C++ programming is required to use the SDK. Register for and download the SDK at **http://www.realnetworks.com/devzone/**.

How This Manual Is Organized

This manual contains the following chapters:

Chapter 1: Embedded Presentations

This chapter provides an overview of embedding RealPlayer presentations on a Web page, with descriptions of the `<EMBED>` tag for plug-ins and the `<OBJECT>` tag for ActiveX controls. A list of the available tag parameters is also provided.

Chapter 2: Using Scripting

This chapter provides a basic overview of scripting for the Embedded RealPlayer controls on your Web page, with emphasis on the methods supplied with the RealPlayer.

Chapter 3: Using Callbacks

This chapter provides an overview of how to use the RealPlayer callback methods.

Chapter 4: Methods

This chapter provides complete descriptions of all the RealPlayer embedded methods.

Chapter 5. Callback Methods

This chapter provides complete descriptions of all the RealPlayer callback methods.

In addition, an appendix, "Compatibility", is included that describes the compatibility of this version of the Embedded RealPlayer methods and callbacks to those of previous versions.

Conventions Used in this Manual

The following table explains the conventions used in this manual.

Notational Conventions

Convention	Meaning
variables	Italicized text represents variables. Substitute values appropriate for your situation.
[options]	Square brackets indicate optional values you may or may not need to use.
choice 1|choice 2	Vertical lines separate values you can choose between.
...	Ellipses indicate nonessential information omitted from the example.

Technical Support

For technical support with RealSystem, please fill out the form at:

http://customerrelations.real.com/scripts/rnforms/contact_tech_service.asp

The information you provide in this form will help technical support personnel to give you a prompt response. For general information about RealNetworks' technical support, visit **http://service.real.com/help/call.html**

RealForum

RealNetworks also encourages you to join RealForum, an e-mail discussion group about RealSystem products where developers and content producers post tips and ask for assistance. RealNetworks employees monitor the postings and offer suggestions as appropriate. You can sign up for RealForum by connecting to **http://realforum.real.com/** and clicking on **New user**.

CHAPTER 1

EMBEDDED PRESENTATIONS

The easiest way to create an Embedded RealPlayer in a Web page is to include the RealPlayer Netscape plug-in or ActiveX control in your HTML application. An embedded presentation created in this manner typically includes an image window and one or more standard RealPlayer controls, such as the **Stop** button, the volume slider, and so on.

> **For More Information:** For more information about creating an embedded RealSystem presentation, see the *RealSystem Production Guide* available at **http://service.real.com/help/library/index.html**.

Playback of the embedded presentation is controlled using the methods described in Appendix A "Methods"and user interaction with the embedded controls is managed using the callback methods defined in Appendix B "Callback Methods". This chapter describes how to include the Netscape plug-in or ActiveX control in your HTML application and provides the basic syntax for calling methods and callback methods from JavaScript or VBScript.

Choosing the Netscape Plug-in or ActiveX Control

To provide Web page playback, RealPlayer includes a plug-in for browsers that support the Netscape plug-in architecture:

- Netscape Navigator 4.0 and higher.

  > **Note:** There are known compatibility issues with some versions of Netscape Navigator 6.0. Be sure to test for compatibility.

- Microsoft Internet Explorer 5.0 and higher.

It also has an ActiveX control that provides playback capabilities within these products:

- Microsoft Internet Explorer 4.0 and higher on Microsoft Windows.
- Most applications that support ActiveX controls, such as Visual Basic and Visual C++.

Because they both have the same capabilities, you can use either the plug-in or the ActiveX control depending on which products you need to support.

Using <EMBED> Tags for the Netscape Plug-in

To use RealPlayer's Netscape plug-in, you add `<EMBED>` tags to your Web page HTML. A typical `<EMBED>` tag has three necessary parameters (`SRC`, `WIDTH`, and `HEIGHT`), that are used to identify your presentation and the dimensions of the playback area. Many other parameters are available, but they are considered to be optional.

> **For More Information:** For a list of all `<EMBED>` parameters, see Appendix C "Tag Parameters" on page 95.

The syntax for a typical `<EMBED>` tag looks like the following:

```
<EMBED SRC="my_content.rpm" WIDTH=300 HEIGHT=134 optional parameter=value>
```

This example tag creates a playback area 300 pixels wide by 134 pixels high within your Web page, and displays the contents of `my_content.rpm` within the playback area.

> **Tip:** The `SRC` parameter may be omitted, when the content mime `TYPE` parameter is specified similar to:
> `<EMBED ..., TYPE="audio/x-pn-realaudio-plugin", ...>`,
> however, doing so may produce unexpected results. Therefore, it is strongly recommended that you always include the `SRC` parameter and at least supply the name of an empty presentation file.

All parameters typically have the form `PARAMETER=value`. The parameter names can be any case, although this manual depicts them in uppercase. Except for file names, that must typically be lowercase, parameter values are not case-sensitive. Unless they are URLs, parameter values do not need to be inside quotation marks.

Using <OBJECT> Tags for the ActiveX Control

To use RealPlayer's ActiveX control, you add `<OBJECT>` tags to your Web page HTML. The tag definition must include the RealPlayer classID value:

```
CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
```

and specify the width and height dimensions of the playback area. When you intend to use scripting to reference the control, you must also specify a value for the `ID` parameter, such as `ID=RVOCX`.

The syntax for a typical `<OBJECT>` tag looks like the following:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
WIDTH=300 HEIGHT=134>
...parameters...
</OBJECT>
```

This example tag creates a playback area 300 pixels wide by 134 pixels high within your Web page. Between `<OBJECT>` and `</OBJECT>` you can define any number of additional parameters using this syntax:

```
<PARAM NAME="name" VALUE="value">
```

`PARAM`, `NAME`, and `VALUE` markers can be any case, although this manual depicts them in uppercase. Except for file names, that must typically be lowercase, parameter values are not case-sensitive. Always enclose parameter values in double quotation marks.

For More Information: For a list of all `<OBJECT>` parameters, see Appendix A "Methods".

Using Scripting to Access Embedded RealPlayer Controls

Using a scripting language such as JavaScript or VBScript, you can access RealPlayer functions such as the stop, play, and volume controls, without using RealPlayer's standard interface components. For example, you can use your own graphic image for a Stop button, capture user interaction with the image, and stop the clip playback.

For More Information: For detailed descriptions about using scripting languages with the RealPlayer controls, see "Chapter 2: Using Methods and Callbacks" beginning on page 17.

Using RealPlayer Methods through JavaScript

To extend RealPlayer's Netscape plug-in functionality with JavaScript, you first embed the source file in an HTML page with the `<EMBED>` tag, for example:

```
<EMBED NAME=javademo
   SRC="demo.rpm"
   WIDTH=220 HEIGHT=180
   CONSOLE=one
   CONTROLS=ImageWindow
   BACKGROUNDCOLOR=white
   CENTER=true
   >
```

In the `<EMBED>` tag, the `NAME` parameter provides the name used by the JavaScript functions. For JavaScript to function with RealPlayer, the `<EMBED>` tag must **not** contain the parameter `NOJAVA=true`. That parameter prevents the browser's Java Virtual Machine from starting up in Netscape version 4.x.

> **Warning!** In Netscape version 4.x, when the parameter `NOJAVA` is set to `true`, scripting is unavailable. Setting the `NOJAVA` parameter in Netscape 6.0 has no effect.

You can then use JavaScript to issue RealPlayer commands to control the embedded presentation. The following example shows a simple form that provides a Play, Pause, and Stop button for the embedded presentation. <a href="../samples/java/playback.htm">Click here</a> to see this presentation.

```
<FORM>
<INPUT TYPE="button" VALUE="Play" onClick="document.javademo.DoPlay()">
<INPUT TYPE="button" VALUE="Pause" onClick="document.javademo.DoPause()">
<INPUT TYPE="button" VALUE="Stop" onClick="document.javademo.DoStop()">
</FORM>
```

To refer to an embedded instance with JavaScript, you include a `NAME` parameter in the `<EMBED>` tag:

```
<EMBED NAME=vid SRC="..." WIDTH=300 HEIGHT=134>
```

You can then refer to the instance through a JavaScript command such as this:

```
<Input Type="button" Value="play" onClick="document.vid.DoPlay()">
```

If you include more than one instance of a single type of embedded control or a variety of different embedded controls in your HTML document, give each instance a unique `NAME`. This ensures that you can use JavaScript to manage each embedded control individually, if necessary.

For More Information: "Appendix A: Methods" beginning on page 41 lists the RealPlayer methods.

Using RealPlayer Methods through VBScript

To extend RealPlayer's ActiveX functionality on Internet Explorer, you first embed the source file in an HTML page with the `<OBJECT>` tag:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
WIDTH=220 HEIGHT=180>
<PARAM NAME="SRC" VALUE="rtsp://realserver.company.com/media/animate.swf">
<PARAM NAME="CONSOLE" VALUE="one">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="BACKGROUNDCOLOR" VALUE="white">
<PARAM NAME="CENTER" VALUE="true">
</OBJECT>
```

In the `<OBJECT>` tag, the `ID` parameter identifies the embedded clip for reference by VBScript parameters.

You can then use VBScript, or any programming language supported by the browser, to issue RealPlayer commands to control the embedded presentation. The following example shows a simple form that provides a Play, Pause, and Stop button for the embedded presentation. <a href="../samples/vb/playback.htm">Click here</a> to see this presentation (this presentation requires Internet Explorer).

```
<FORM>
<input TYPE="button" VALUE="Play" NAME="doplay">
  <script LANGUAGE="VBScript" FOR="doplay" EVENT="onClick">
  RVOCX.DoPlay
  </script>
<input TYPE="button" VALUE="Pause" NAME="pause">
  <script LANGUAGE="VBScript" FOR="pause" EVENT="onClick">
  RVOCX.DoPause
  </script>
<input TYPE="button" VALUE="Stop" NAME="stop">
  <script LANGUAGE="VBScript" FOR="stop" EVENT="onClick">
  RVOCX.DoStop
  </script>
</FORM>
```

For More Information: "Appendix A: Methods" beginning on page 41, lists the RealPlayer methods.

Handling Callback Events From RealPlayer

RealPlayer reports the events that occur within the application to the Netscape plug-in or ActiveX control using a set of predefined API methods known as *callbacks*. You can use these callback methods on your Web page to intercept and interpret RealPlayer events, including tracking mouse movement, capturing user interactions with the application controls, or monitoring the progress of your presentation.

For More Information: "Appendix B: Callback Methods" beginning on page 81, lists the RealPlayer callback events.

COMMENT: **The information in this chapter is too sparse.
I need a list of additional topics to add. Any Suggestions?**

Receiving Callbacks Through JavaScript

RealPlayer communicates the events that occur in a Netscape plug-in via a set of internal callback routines. However, depending on the platform and version of Netscape you are targeting, the Embedded RealPlayer supports the handling of these events using different mechanisms.

If you are developing a plug-in for Netscape version 6.0 running on Windows, UNIX, or Macintosh, the mechanism consists of including a new `<EMBED>` tag in your JavaScript and specifying which events to receive. When targeting older versions of Netscape, LiveConnect should continue to be used to receive the callbacks.

The following sections describe the different event handling mechanisms in more detail.

Handling Events in Netscape Version 6.0

Netscape® version 6.0 does not support callback event handling in the same manner as previous versions. For this reason, Embedded RealPlayer build number 6.0.8.1024 and later introduces a new mechanism for event handling involving the use of JavaScript and callback methods in a Netscape plug-in.

The procedure is available for development using JavaScript and in the following configurations:

Supported Configurations for the New Event Handling Mechanism

Web Browser	Version	Windows	Macintosh	UNIX
Netscape	6.0 or later	yes	yes	no*
Internet Explorer	all versions	no	yes	no*

*Will be available in a post-version 8 release of the Unix RealPlayer.

Note: The new event handling mechanism is not available for development in C++ or for use by an ActiveX control.

To use the new mechanism in your JavaScript plug-in, add the `SCRIPTCALLBACKS` parameter to your `<EMBED>` tag defintion. Identify the events to handle by providing a comma-separated list of callback methods or by specifying ‘`All`’ to capture all events. For example:

```
<EMBED SCRIPTCALLBACKS=OnPresentationOpened,
                         OnPositionChange,OnPresentationClosed>
```

- or -

```
<EMBED SCRIPTCALLBACKS=All>
```

You don’t need to worry about the backward compatibility of the new tag because all versions of Embedded RealPlayer ignore unrecognized tags. However, you may detect the RealPlayer version and if earlier than build 6.0.8.1024, inform the user that their RealPlayer version does not support the new event handling features.

Handling Events in Netscape Version 4.x

When developing for Netscape versions 4.x, you must use LiveConnect to receive the callbacks sent by the RealPlayer. LiveConnect is described at **http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/**.

To receive the RealPlayer callbacks, you must first embed a Java `<APPLET>` tag in your HTML code. This tag should include a reference to the RealPlayer event interface class file, for example (`CODE="RMObserver.class"`), and have the `MAYSCRIPT` attribute set. In addition you must also provide a `NAME` attribute (such as `NAME="YourNameHere"`) that will be used in JavaScript to identify the instance of the applet. The following example shows how to use the `<APPLET>` tag:

```
<APPLET CODE="RMObserver.class" WIDTH=1 HEIGHT=1 NAME="YourNameHere"
MAYSCRIPT>
</APPLET>
```

You can then use the name provided in the `<APPLET>` tag to receive callbacks. For example, you could use the following line to determine when a clip has closed:

```
if(document.YourNameHere.OnClipClosed())
```

On Linux, the `RMObserver.class` file is found in the `raplayer.zip` file in the `/RealPlayer9` directory. On Windows, this class file is found in the `rpcl3260.zip` file in the `\Program Files\Netscape\Communicator\Program\Plugins` directory.

COMMENT: The previous paragraph needs to be expanded to include all flavors of Unix, the Mac, and any change to the Windows class file name and location. Should also include the location of the rpcl3260.zip file in Windows if, for instance, you are only using JScript in Internet Explorer.

To provide backward compatibility, the RealPlayer installation includes the following classes for event notification:

- `RMObserver.class`

  RMObserver is a Java interface for events coming from RealPlayer 7 or later. Any object implementing this interface can register itself into RealPlayer 7 or later, to get the full set of RealPlayer callback notifications.

- `G2Observer.class`

  G2Observer is a Java interface for events coming from RealPlayer G2. Any object implementing this interface can register itself into RealPlayer G2 to get the set of RealPlayer G2 callback notifications.

- `RAObserver.class`

  RAObserver is a Java interface for events coming from RealPlayer 5.0. Any object implementing this interface can register itself into RealPlayer 5.0 to get the set of RealPlayer 5.0 callback notifications.

Receiving Callbacks Through VBScript

To receive callbacks through VBScript, you use the `<OBJECT>` tag ID, shown here set to `RVOCX`:

```
<OBJECT ID=RVOCX HEIGHT=256 WIDTH=256>
  CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
<PARAM NAME="controls" VALUE="all">
<PARAM NAME="SRC" VALUE="http://www.company.com/sample.rpm">
</OBJECT>
```

You then use a `<SCRIPT>` tag to receive a VBScript callback. The following example shows a callback for `OnShowStatus`:

```
<P>
Status Text:
<input type="text" name="statusText" size=100><br>
</P>
<SCRIPT language="VBS">
Sub RVOCX_OnShowStatus(byVal text)
  statusText.Value=text
End Sub
</SCRIPT>
```

Sample Files

<a href="../samples/java/Open3.htm">Javascript samples</a>

<a href="../samples/vb/PLAYBACK.HTM">VB Playback sample</a>

CHAPTER

2 USING METHODS AND CALLBACKS

The Embedded RealPlayer controls have methods that can be called to control the playback of your embedded presentations and callback methods that can be used to handle events sent from the RealPlayer. This chapter provides a brief introduction to the methods and callbacks and describes how groups of related methods can be used together to enhance your embedded presentations.

Categories of Methods

The methods listed in Appendix A "Methods" are described here according to the usage category to which they belong. For example, all of the methods related to enabling application authors to determine the current playback status and to control playback of their presentation, are grouped together and described in the following section, "Controlling Playback".

Note: Obsolete methods are included in these categories, but are colored gray. Read the obsolete method's description to determine which method to use instead.

Controlling Playback

Play, pause, and stop the clip using your own controls or determine the clip playback status.

CanPlay	CanPause	CanStop	CanPlayPause
DoPlay	DoPause	DoStop	DoPlayPause

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `playback.htm` or `playbac1.htm` sample application.

If you want to supply your own play, pause, and stop controls for your embedded player instead of using the built-in controls, use the `DoPlay`, `DoPause`, and `DoStop` playback methods. The following example shows one simple way to add your own controls:

```
<INPUT TYPE="button" VALUE="Play" onClick="document.javademo.DoPlay()">
<INPUT TYPE="button" VALUE="Pause" onClick="document.javademo.DoPause()">
<INPUT TYPE="button" VALUE="Stop" onClick="document.javademo.DoStop()">
```

This category of methods can also be used to determine the playback status of the clip. The `CanPlay`, `CanPause`, and `CanStop` playback methods indicate whether the clip can be played, paused, or stopped. For example, if `CanPlay` returns `true`, it means the clip is either paused or stopped and the clip is ready to play. If `CanPlay` returns `false`, it means the clip is already playing and is not yet ready to begin playing again.

Obtaining Play State Information

Get the current play state of a presentation or determine the elapsed and remaining buffering times of a presentation.

GetBufferingTimeElapsed	Netscape OnlyGetLastMessage	GetPlayState
GetBufferingTimeRemaining	GetLastStatus	

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `state.htm` sample application.

RealPlayer provides a method for obtaining the current play state of a presentation on your Web page. In addition, you can determine the elapsed buffering time and remaining buffering time of a presentation.

`GetPlayState` returns an integer that indicates the current state of the RealPlayer. The possible play states denoted by the returned integer include:

- 0 — Stopped

- 1 — Contacting
- 2 — Buffering
- 3 — Playing
- 4 — Paused
- 5 — Seeking

Knowledge of the current presentation state can be very useful. For example, you could disable the image status text written along the bottom of the image window, and replace it with the current presentation state in a custom display panel.

> **Note:** The image status text only appears along the bottom of the image window when no other control, such as a status panel or status field, is available to display the status text.

Like the information returned from `GetPlayState`, the information displayed along the bottom of the image window indicates whether the playback is contacting, buffering, and so on. To disable this text, set the `SetImageStatus` method's parameter to `false`. To determine whether the status text is currently enabled or disabled, use the `GetImageStatus` method. In addition,

Another example is to determine the amount of time that has elapsed since buffering began for a requested presentation. After the embedded player has contacted the host for the requested presentation, the player buffers the content to provide smooth playback of the presentation. You can obtain the elapsed time using the `GetBufferingTimeElapsed` method or determine the amount of buffering time remaining, using the `GetBufferingTimeRemaining` method.

Specifying Control Attributes

Get or set the current state of the embedded controls on your Web page.

> **Note:** For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `prefetch.htm`, `controls.htm`, or `console.htm` sample application.

The attribute methods supply information about the current state of the embedded controls on your Web page, along with a means of dynamically changing the state of these controls. (These methods can also be used in programming languages, such as C++, to set the control's parameters.)

RealPlayer provides a series of "set" methods with which you can change the attributes of the player embedded on your Web page. Many of these "set" methods mirror the parameters used with the `<EMBED>` or `<OBJECT>` tag, and dynamically change the state of an embedded control. For example, `SetBackgroundColor` is the equivalent of the `BACKGROUNDCOLOR` parameter of the `<EMBED>` or `<OBJECT>` tag. You can use `SetBackgroundColor` to dynamically change the background color of an `ImageWindow` at any time.

The following methods are associated with the parameters used with the `<EMBED>` or `<OBJECT>` tag:

SetAutoGoToURL	AUTOGOTOURL	SetLoop	LOOP
SetAutoStart	AUTOSTART	SetNoLabels	NOLABELS
SetBackgroundColor	BACKGROUNDCOLOR	SetNoLogo	NOLOGO
SetCenter	CENTER	SetNumLoop	NUMLOOP
SetConsole	CONSOLE	SetPreFetch	PREFETCH
SetControls	CONTROLS	SetShuffle	SHUFFLE
SetMaintainAspect	MAINTAINASPECT	SetSource	SRC
SetConsoleName	- - -	SetCanSeek	- - -

Other methods let you "get" the current state of the attributes of the player embedded on your Web page. For example, `GetBackgroundColor` returns a hexadecimal value for the current background color.

DoGotoURL	GetCanSeek	GetAutoStart	GetNumLoop
GetAutoGoToURL	GetCenter	GetLoop	GetPreFetch
GetBackgroundColor	GetConsole	GetNoLabels	GetShuffle
GetMaintainAspect	GetControls	GetNoLogo	GetSource

Seeking Through a Clip

Get the current clip position, total clip length, and determine if you can seek through the presentation.

`GetLength`	`GetPosition`	`SetPosition`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `position.htm` sample application.

The Embedded RealPlayer provides several methods that can be used to obtain information about the currently-playing clip, including determining the present clip position, total length, and whether you can seek through the presentation.

During playback, you can use the `GetCanSeek` method to verify whether you can seek through the clip. When the method returns `true`, you can then use the and `GetPosition` methods to obtain the clip length and current position in the clip. The actual length is returned as the total number of milliseconds in the clip. The position is returned as the number of milliseconds already played. You can also use the information returned from the `GetLength` method to ensure that the postion entered in the `SetPosition` method does not exceed the total length of the clip.

Another method that can be used when the `GetCanSeek` method returns `true` is `SetCanSeek`. This method allows you to specify whether a clip can be seeked. However, if you use the `SetCanSeek` method to allow seeking through an inherently unseekable clip, such as a live broadcast, the set method will have no effect.

Accessing Clip Title, Author, and Copyright Information

Get or set the title, author, and copyright statement for a clip.

`GetTitle`	`GetAuthor`	`GetCopyright`	`GetClipHeight`
`SetTitle`	`Netscape OnlyNetscape OnlySetAuthor`	`SetCopyright`	`GetClipWidth`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `cliptac.htm` sample application.

When a title, author, and copyright statement have been encoded in a clip, this information can be retrieved using the `GetTitle`, `GetAuthor`, and `GetCopyright` methods. In multi-clip presentations, these methods return the information associated with the currently-playing clip.

The `SetTitle`, `Netscape OnlyNetscape OnlySetAuthor`, and `SetCopyright` methods can be used to dynamically change whether the title, author, and copyright information for a clip is displayed. This information is normally displayed in the clip information control (the `www.real.com Home Button` control) for each clip. In multi-clip presentations, these methods override the clip information for the entire presentation, not just the currently-playing clip. These methods are useful if you are presenting a multi-clip presentation and want to supply your own title, author, and copyright information for the entire presentation.

Directing a Playlist in a Multi-clip Presentation

Get the playlist information about each clip in a presentation and use it to move from clip to clip in the playlist, or acquire information about any clip.

`DoNextEntry`	`HasNextEntry`	`GetEntryAbstract`	`GetCurrentEntry`
`DoNextItem`	`HasNextItem`	`GetEntryAuthor`	`GetNumEntries`
`DoPrevEntry`	`HasPrevEntry`	`GetEntryCopyright`	
`DoPrevItem`	`HasPrevItem`	`GetEntryTitle`	

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `playlist.htm` sample application.

Multi-clip presentations are made up of playlists that contain useful information about each of the clips in the presentation. You can use this data to move from clip to clip in the playlist, or acquire content information about any clip in the presentation.

To get the number of entries in the playlist, use the `GetNumEntries` method. In this method, a single entry is always returned as the number 1. You can also retrieve the entry number of the clip that is currently playing using the `GetCurrentEntry` method. Note, however, that this method begins counting at zero, therefore the third entry in a playlist would return the number 2. While playing a multi-clip presentation, use the `HasNextEntry` method to determine if there is another entry in the playlist. You can then use the `DoNextEntry` method to jump to the next clip. Use the `HasPrevEntry` method to determine if a previous entry exists in the playlist. If so, you can then use the `DoPrevEntry` method to jump back to the beginning of the previous clip.

After you have determined the number of entries in a playlist, you can use the `GetEntryAbstract`, `GetEntryAuthor`, `GetEntryCopyright`, and `GetEntryTitle` methods to retrieve the abstract, author, copyright, and title information for any clip in the presentation. Simply supply the number of the playlist entry for which you want information as the parameter for these methods. These methods return a string that contains the information for the specified entry.

Determining Live Broadcast

Determine if the currently playing clip is a live presentation.

GetLiveState

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `liveplay.htm` sample application.

The `GetLiveState` method indicates whether or not the current clip is a live presentation. This information could be useful, for instance, if you wanted to know whether to use the clip positioning methods. In this case, if `GetLiveState` returns `true`, that means the current clip is part of a live presentation, and you cannot use the clip positioning methods.

Display User Interface Dialogs

Use RealPlayer menus to set RealPlayer preferences, view playback statistics, or view information about the RealPlayer.

AboutBox	GetShowStatistics	HideShowStatistics	EditPreferences

ActiveX OnlyGetShowAbout	SetShowStatistics	Netscape OnlyIsStatisticsVisible	GetShowPreferences
ActiveX OnlySetShowAbout			SetShowPreferences

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `playlist.htm` sample application.

RealPlayer contains several menus that can be used to set various player preferences, view the playback statistics, and view information about the player.

You can access the player preferences using the `SetShowPreferences` method. Setting this method's parameter to `true` brings up an abbreviated version of the standard RealPlayer Preferences dialog box. To determine if the Preferences dialog box is already being displayed, use the `GetShowPreferences` method. If this method returns `true`, the Preferences dialog box is already being displayed.

To view the playback statistics, use the `SetShowStatistics` method. Setting this methods's parameter to `true` brings up the standard RealPlayer Statistics dialog box. To determine if the Statistics dialog box is already being displayed, use the `GetShowStatistics` method. If this method returns `true`, the Statistics dialog box is already being displayed.

RealPlayer's About dialog box contains information about the version of RealPlayer and its individual components being used in your embedded Web Page. To view the About dialog box, use the `ActiveX OnlySetShowAbout` method. Setting this method's parameter to `true` brings up the RealPlayer About dialog box. To determine if the About dialog box is already being displayed, use the `ActiveX OnlyGetShowAbout` method. If this method returns `true`, the About dialog box is already being displayed.

Error Handling

Specify whether errors detected by the RealPlayer should be trapped and displayed.

`GetWantErrors`	`GetLastStatus`	`GetLastErrorUserCode`	`GetLastErrorRMACode`
`SetWantErrors`	`GetLastErrorSeverity`	`GetLastErrorUserString`	`GetLastErrorMoreInfoURL`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `errors.htm` sample application.

You can designate whether or not the RealPlayer embedded in your Web page displays error dialogs. If the `SetWantErrors` method is set to `true`, error messages from the player are trapped, and not displayed in an error dialog box. If `SetWantErrors` is set to `false`, error messages are displayed in an error dialog box. You can use the `GetWantErrors` method to determine if error messages are being trapped. If `GetWantErrors` returns `true`, error messages are being trapped. If `GetWantErrors` returns `true`, error message will be displayed when they occur.

COMMENT: What, exactly do the GetEnableMessageBox and SetEnableMessageBox methods do? What is their relationship to Set/GetWantErrors?

COMMENT: Answer from Haydon: Defines whether or not ANY dialogs are presented?

When a player error occurs, the `GetLastErrorMoreInfoURL` method supplies a URL that can be used to provide more information than is provided by the standard error string.

COMMENT: <<But how does one go about setting this URL - need an example>>

COMMENT: Answer from Haydon: URL is part of the error message that is returned ("GetLastErrorMoreInfo" URL)

If no URL is available to supply more information, this method returns nothing.

The `GetLastErrorRMACode` method returns a value that represents the error code returned by RealPlayer. The values of the error codes and their descriptions can be found in the RealSystem SDK `rmaerror.h` file and in the *RealSystem SDK Developer's Guide*.

COMMENT: GetLastErrorRMAString has been removed.

Other RealPlayer error methods include `GetLastErrorSeverity`, which responds with the severity of the error, and `GetLastStatus`, which responds with the text of the last status message returned by the `OnShowStatus` callback.

Also included are two user-defined error methods: `GetLastErrorUserCode` and `GetLastErrorUserString`. The `GetLastErrorUserCode` returns a user-defined error code, and `GetLastErrorUserString` returns a user-defined string that describes the last error that occurred. If you have designed a custom plug-in for RealPlayer, you can use both of these methods to return your own error codes and error strings for your plug-in. If you are not providing a custom plug-in for RealPlayer, both of these methods return nothing.

Setting the Display Size

.....

GetDoubleSize	GetFullScreen	GetOriginalSize
SetDoubleSize	SetFullScreen	SetOriginalSize

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `display.htm` or `context.htm` sample application.

You can change the size of the image window on your Web page to either the original size you specified in the `<EMBED>` or `<OBJECT>` tag, or to full screen.

To set the image window to full screen, use the `SetFullScreen` method with its parameter set to `true`. To determine if the image window is set to full screen, use the `GetFullScreen` method. If this method returns `true`, the image window is set to full screen.

To set an image window to the original size specified in the `<EMBED>` or `<OBJECT>` tag on your Web page, use the `SetOriginalSize` method with its parameter set to `true`. To determine if the image window is currently set to its original size, use the `GetOriginalSize` method. If this method returns `true`, the image window is set to its original size.

The `SetDoubleSize` and `GetDoubleSize` methods are intended for advanced applications only. They have no effect within a Web page.

Controlling Audio

.....

GetMute	Netscape OnlyGetVolume	GetStereoState
SetMute	Netscape OnlySetVolume	

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `display.htm` or `context.htm` sample application.

You can embed various audio controls in your Web page to get and set the volume of the presentation, mute the presentation, and determine if the presentation is being played in stereo or monaural.

To set the volume of the presentation, use the `Netscape OnlySetVolume` method. The parameter you enter for this method is a percentage of the total volume, that is, if you want the volume to be set to 50%, use the number 50 as this method's parameter. To determine the current volume of the presentation, use the `Netscape OnlyGetVolume` method. This method returns the current volume setting as a percentage of the total volume.

To mute the presentation, use the `SetMute` method. Set this method's parameter to `true` to mute the presentation, and `false` to return the sound volume to its previous level. To determine if the presentation is currently muted, use the `GetMute` method. This method returns `true` if the presentation is muted and `false` if it is not.

To determine if the current presentation is stereo or monaural, use the `GetStereoState` method. This method returns `true` if the presentation is stereo, and `false` if the presentation is monaural.

Controlling the Pop-up Context Menu

.....

`GetEnableContextMenu`	`SetEnableContextMenu`
`GetEnableDoubleSize`	`SetEnableDoubleSize`
`GetEnableFullScreen`	`SetEnableFullScreen`
`GetEnableOriginalSize`	`SetEnableOriginalSize`
`GetImageStatus`	`SetImageStatus`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `display.htm`, `state.htm`, or `context.htm` sample application.

Every embedded control you include on your Web page contains a context menu that can be accessed by placing the mouse cursor over the control, then clicking the right button, or equivalent. This context menu contains selections to play, stop, or mute the presentation in the image window, along with selections that contain clip information in the presentation, zoom controls, access to the Preference and Statistics dialog boxes, and information about the player and presentation.

In some cases, you may have more than one set of `<EMBED>` or `<OBJECT>` tags on your Web page. For example, you could have two `ImageWindow` controls, each of which has its own unique play, pause, and stop button. Each set of controls would then have a unique `CONSOLE` parameter, one for the first `ImageWindow` and another for the second `ImageWindow`. All controls linked by the first `CONSOLE` name interact with each other, but not with the other set of controls on the Web page, which has a different `CONSOLE` name. Therefore, when you right-click on one of these controls to bring up the context menu, that menu only relates to the set linked by the `CONSOLE` name for the control on which you clicked.

The context menu is normally enabled. To disable the context menu so that it does not appear when the right mouse button is pressed, use the `SetEnableContextMenu` method. By setting this method's parameter to `false`, the context menu is disabled. To reenable the context menu, set this method's parameter to `true`. To determine whether the context menu is currently enabled or disabled, use the `GetEnableContextMenu` method.

You can also seperately enable or disable the zoom selections in the context menu. This may be useful if you want to prevent someone from using these menu selections on your presentation. You can disable each of the zoom selections using the `SetEnableOriginalSize`, `SetEnableDoubleSize`, and `SetEnableFullScreen` methods. If you set these method's parameters to `false`, the corresponding menu selection in the context menu is disabled and the selection is grayed. To reenable these menu selections, set the method's parameter to `true`. To determine the current setting of these menu selections, use the `GetEnableOriginalSize`, `GetEnableDoubleSize`, and `GetEnableFullScreen` methods.

Getting Network Information

.....

GetSourceTransport	GetPacketsTotal	GetPacketsMissing	GetBandwidthAverage
GetNumSources	GetPacketsReceived	GetPacketsOutOfOrder	GetBandwidthCurrent
	GetPacketsEarly	GetPacketsLate	

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `network.htm` sample application.

You can obtain information from a user's RealPlayer to determine how your Web page reacts to the user's network capabilities.

Two methods provide playback bandwidth information from the user's RealPlayer. The first, `GetBandwidthAverage`, returns the average playback bandwidth of the user's RealPlayer, in bits per second, from the beginning of playback to the time when this method is called. The second, `GetBandwidthCurrent`, returns the current playback bandwidth in bits per second.

The `GetNumSources` method returns the number of sources in the presentation, where a source is a piece of media or an entry in a playlist or .rpm file.

Several methods are provided that let your Web page monitor the network activity of the user's RealPlayer. `GetPacketsReceived` indicates the total number of packets that the player received correctly from the RealSystem Server, and `GetPacketsTotal` indicates the total number of packets that should have been received by the player up to the point at which this method was called, including the packets that were lost. If no packets were lost, `GetPacketsTotal` returns the same number as `GetPacketsReceived`.

You can also monitor how well the user's RealPlayer is receiving packets. `GetPacketsEarly` indicates the number of packets that were received early. `GetPacketsLate` indicates the number of packets that were received late. `GetPacketsMissing` indicates the number of packets that were never received. `GetPacketsOutOfOrder` indicates the number of packets that were received out of order.

`GetSourceTransport` returns the type of protocol used for the stream by the user's RealPlayer. This method either returns "udp" or "tcp", depending on the source transport.

COMMENT: **<<Is this correct? Does anything more need to be said? Also note that if the file is local, this method returns nothing. Does that need to be said here?>>**

Obtaining RealPlayer Version Information

.....

`GetIsPlus`	`GetVersionInfo`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `version.htm` sample application.

Two methods provide version information for the RealPlayer controls you have embedded on your web page.

The first, `GetIsPlus`, returns `true` if the version of the player playing back your presentation is RealPlayer Plus or `false` if the player is the basic RealPlayer. With this information, you could provide different options to the different types of players. Since the basic RealPlayer does not provide all of the options of RealPlayer Plus, your Web page could then respond differently to a user playing your presentation with RealPlayer Plus.

The second method that supplies version information is `GetVersionInfo`. This method returns the version of the RealPlayer plug-in running on the user's machine (in period-deliniated form, such as "6.0.7.788", which is the Embedded RealPlayer build for RealPlayer 8). With this information, your Web page can once again provide different options for different versions of the player.

The following table lists the equivalent RealPlayer releases, stand-alone builds, and embedded player builds. Use this information to differentiate between versions.

Version Capatiblity Table

RealPlayer Release Version	Standalone Build Number	Embedded RealPlayer Build Number
8 Update 3	6.09.584	6.0.8.1024
8 Update 2	6.0.9.450	6.0.7.881
8 Gold	6.0.9.357	6.0.7.788
7 Update 1	6.0.8.122	6.0.7.529
7 Gold	6.0.7.380	6.0.7.407
G2 Update 3	6.0.6.99	6.0.6.98
G2 Update 2	6.0.6.33	6.0.6.33
G2 Update 1	6.0.5.27	6.0.5.27
G2 Gold	6.0.3.128	6.0.6.131

Event Handling

.....

`GetConsoleEvents`	`GetWantKeyboardEvents`	`GetWantMouseEvents`
`SetConsoleEvents`	`SetWantKeyboardEvents`	`SetWantMouseEvents`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `events.htm` sample application.

RealPlayer includes methods for enabling or disabling keyboard and mouse events on your Web page. Also included is a method that can be used in plug-ins to make event handling act more like event handling in ActiveX controls.

Available Methods

The `SetWantKeyboardEvents` method sets whether or not keyboard events are returned from the user's RealPlayer. If `SetWantKeyboardEvents` is set to `true`, the keyboard event callbacks, `OnKeyDown`, `OnKeyPress`, and `OnKeyUp`, are sent from RealPlayer when a keyboard event occurs. If this method is set to `false`, keyboard events are not sent. You can use the `GetWantKeyboardEvents` method to determine if keyboard events have been requested from the user's RealPlayer.

The `SetWantMouseEvents` method works in much the same way. If `SetWantMouseEvents` is set to `true`, the mouse event callbacks, `OnLButtonDown`, `OnLButtonUp`, `OnMouseMove`, `OnRButtonDown`, and `OnRButtonUp`, are sent from RealPlayer when a mouse event occurs. If this method is set to `false`, mouse events are not sent. You can use the `GetWantMouseEvents` method to determine if mouse events have been requested from the user's RealPlayer.

The `SetConsoleEvents` method is provided to let plug-in designers make their plug-ins behave more like the ActiveX control. By setting the `value` parameter of the `SetConsoleEvents` method to `false`, only the plug-in to which the console connects will send events.

How Event Handling Works

In ActiveX, you specify the particular ActiveX control for which your function is handling events. Thus any event that occurs in reference to a specific ActiveX control is always handled only by that single control.

If, however, you are using multiple plug-ins on a page and an event occurs on that page, a Java applet cannot determine from which plug-in the event was sent. (Although you could add multiple Java applets to the page to handle each individual plug-in, in general all events for all plug-ins that share the same console name are sent to a single applet.) For example, if you have two plug-ins on a page and generate a mouse or button event in either of the plug-ins, the event goes to the applet, but the applet cannot determine which plug-in generated the event.

The `SetConsoleEvents` method does not affect ActiveX controls.

You can use the `GetConsoleEvents` method to determine whether console events are enabled. If `GetConsoleEvents` is `true`, then console events are enabled and events from any applet will be sent to the plug-in. If this method is set to `false`, console events are disabled and only the plug-in to which the console connects will send events.

Getting User Preference Settings

.....

`GetConnectionBandwidth`	`GetUserCountryID`
`GetPreferredLanguageID`	`GetPreferredLanguageString`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `prefer.htm` sample application.

Some of the preference settings from the RealPlayer Preferences dialog can be retrieved by your Web page. You can then use this information to customize your Web page to the preferences of the user's RealPlayer.

The `GetConnectionBandwidth` method returns the value of the user's RealPlayer connection bandwidth preference in milliseconds. You can use this information to determine whether your presentation will play properly on the user's RealPlayer (in the case of a single bandwidth presentation) or to determine the optimal playback bandwith (in the case of SureStream content).

The `GetPreferredLanguageString` method returns a string that identifies the preferred language for content as specified by the user under the Content tab of her RealPlayer Preferences dialog. The user's RealPlayer returns a string of the form "*xx-yy, xx, **". You can use the "*xx-yy*" portion of the string to determine the user's preferred language. For example, English (United States) returns the raw string "en-us, en, *", of which "en-us" refers to the preferred language. A complete list of all the language preferences that can be set by

RealPlayer can be found in the SMIL language codes appendix of the *RealSystem Production Guide*.

> **Note:** Some versions of RealPlayer return portions of the preferred language string as capital letters. You should design your Web page to handle these capital letters appropriately when comparing these returned stings against a table of known languages.

`GetPreferredLanguageID` and `GetUserCountryID` return standardized language identifications and standard country identifications. In Windows, these returned values are defined in `olenls.h`.

COMMENT: <<What about in other operating systems?>>

Categories of Callback Methods

The methods listed in Appendix B "Callback Methods" are described here according to the usage category to which they belong. For example, all of the methods related to detecting changes to a presentation, are grouped together and described in the following section, "Detecting Changes in Presentation Data".

> **Note:** Obsolete callback methods are included in these categories, but are colored gray. Read the obsolete method's description to determine which method to use instead.

Detecting Changes in Presentation Data

.....

`OnTitleChange`	`OnAuthorChange`	`OnCopyrightChange`

When a title, author, and copyright statement have been encoded in a clip, this information can be retrieved using the `GetTitle`, `GetAuthor`, and `GetCopyright` methods. In multi-clip presentations, these methods return the information associated with the currently-playing clip.

The `SetTitle`, `Netscape OnlyNetscape OnlySetAuthor`, and `SetCopyright` methods can be used to dynamically change whether the title, author, and copyright information for a clip is displayed. This information is normally displayed in the clip information control (the `www.real.com Home Button` control) for each clip. In multi-clip presentations, these methods override the clip information for the entire presentation, not just the currently-playing clip. These methods are useful if you are presenting a multi-clip presentation and want to supply your own title, author, and copyright information for the entire presentation.

Event Handling

.....

`OnKeyDown`	`OnKeyPress`	`OnKeyUp`	`OnMouseMove`
`OnLButtonDown`	`OnLButtonUp`	`OnRButtonDown`	`OnRButtonUp`

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `events.htm` sample application.

RealPlayer includes methods for enabling or disabling keyboard and mouse events on your Web page. Also included is a method that can be used in plug-ins to make event handling act more like event handling in ActiveX controls.

Available Methods

The `SetWantKeyboardEvents` method sets whether or not keyboard events are returned from the user's RealPlayer. If `SetWantKeyboardEvents` is set to `true`, the keyboard event callbacks, `OnKeyDown`, `OnKeyPress`, and `OnKeyUp`, are sent from RealPlayer when a keyboard event occurs. If this method is set to `false`, keyboard events are not sent. You can use the `GetWantKeyboardEvents` method to determine if keyboard events have been requested from the user's RealPlayer.

The `SetWantMouseEvents` method works in much the same way. If `SetWantMouseEvents` is set to `true`, the mouse event callbacks, `OnLButtonDown`, `OnLButtonUp`, `OnMouseMove`, `OnRButtonDown`, and `OnRButtonUp`, are sent from RealPlayer when a mouse event occurs. If this method is set to `false`, mouse events are not sent. You can use the `GetWantMouseEvents` method to determine if mouse events have been requested from the user's RealPlayer.

The `SetConsoleEvents` method is provided to let plug-in designers make their plug-ins behave more like the ActiveX control. By setting the `value` parameter of the `SetConsoleEvents` method to `false`, only the plug-in to which the console connects will send events.

How Event Handling Works

In ActiveX, you specify the particular ActiveX control for which your function is handling events. Thus any event that occurs in reference to a specific ActiveX control is always handled only by that single control.

If, however, you are using multiple plug-ins on a page and an event occurs on that page, a Java applet cannot determine from which plug-in the event was sent. (Although you could add multiple Java applets to the page to handle each individual plug-in, in general all events for all plug-ins that share the same console name are sent to a single applet.) For example, if you have two plug-ins on a page and generate a mouse or button event in either of the plug-ins, the event goes to the applet, but the applet cannot determine which plug-in generated the event.

The `SetConsoleEvents` method does not affect ActiveX controls.

You can use the `GetConsoleEvents` method to determine whether console events are enabled. If `GetConsoleEvents` is `true`, then console events are enabled and events from any applet will be sent to the plug-in. If this method is set to `false`, console events are disabled and only the plug-in to which the console connects will send events.

Controlling Audio

.....

`OnMuteChange`	`OnVolumeChange`

You can embed various audio controls in your Web page to get and set the volume of the presentation, mute the presentation, and determine if the presentation is being played in stereo or monaural.

To set the volume of the presentation, use the `Netscape OnlySetVolume` method. The parameter you enter for this method is a percentage of the total volume, that is, if you want the volume to be set to 50%, use the number 50 as this method's parameter. To determine the current volume of the presentation, use the `Netscape OnlyGetVolume` method. This method returns the current volume setting as a percentage of the total volume.

To mute the presentation, use the `SetMute` method. Set this method's parameter to `true` to mute the presentation, and `false` to return the sound volume to its previous level. To determine if the presentation is currently muted, use the `GetMute` method. This method returns `true` if the presentation is muted and `false` if it is not.

To determine if the current presentation is stereo or monaural, use the `GetStereoState` method. This method returns `true` if the presentation is stereo, and `false` if the presentation is monaural.

Obtaining Play State Information

.....

`OnGotoURL`	`OnPositionChange`	`OnPosLength`	`OnPlayStateChange`
`OnContacting`	`OnPresentationOpened`	`OnClipOpened`	`OnStateChange`
`OnBuffering`	`OnPresentationClosed`	`OnClipClosed`	`OnShowStatus`
`OnPreFetchComplete`			

For More Information: For a demonstration of how to use one or more of these methods in a Netscape plug-in, see the `prefetch.htm` sample application.

RealPlayer provides a method for obtaining the current play state of a presentation on your Web page. In addition, you can determine the elapsed buffering time and remaining buffering time of a presentation.

`GetPlayState` returns an integer that indicates the current state of the RealPlayer. The possible play states denoted by the returned integer include:

- 0 — Stopped
- 1 — Contacting
- 2 — Buffering
- 3 — Playing

- 4 — Paused
- 5 — Seeking

Knowledge of the current presentation state can be very useful. For example, you could disable the image status text written along the bottom of the image window, and replace it with the current presentation state in a custom display panel.

> **Note:** The image status text only appears along the bottom of the image window when no other control, such as a status panel or status field, is available to display the status text.

Like the information returned from `GetPlayState`, the information displayed along the bottom of the image window indicates whether the playback is contacting, buffering, and so on. To disable this text, set the `SetImageStatus` method's parameter to `false`. To determine whether the status text is currently enabled or disabled, use the `GetImageStatus` method. In addition,

Another example is to determine the amount of time that has elapsed since buffering began for a requested presentation. After the embedded player has contacted the host for the requested presentation, the player buffers the content to provide smooth playback of the presentation. You can obtain the elapsed time using the `GetBufferingTimeElapsed` method or determine the amount of buffering time remaining, using the `GetBufferingTimeRemaining` method.

Error Handling

.....

```
OnErrorMessage
```

You can designate whether or not the RealPlayer embedded in your Web page displays error dialogs. If the `SetWantErrors` method is set to `true`, error messages from the player are trapped, and not displayed in an error dialog box. If `SetWantErrors` is set to `false`, error messages are displayed in an error dialog box. You can use the `GetWantErrors` method to determine if error messages are being trapped. If `GetWantErrors` returns `true`, error messages are being trapped. If `GetWantErrors` returns `true`, error message will be displayed when they occur.

COMMENT: What, exactly do the GetEnableMessageBox and SetEnableMessageBox methods do? What is their relationship to Set/GetWantErrors?

When a player error occurs, the `GetLastErrorMoreInfoURL` method supplies a URL that can be used to provide more information than is provided by the standard error string.

COMMENT: <<But how does one go about setting this URL - need an example>>

If no URL is available to supply more information, this method returns nothing.

The `GetLastErrorRMACode` method returns a value that represents the error code returned by RealPlayer. The values of the error codes and their descriptions can be found in the RealSystem SDK `rmaerror.h` file and in the *RealSystem SDK Developer's Guide*.

COMMENT: GetLastErrorRMAString has been removed.

Other RealPlayer error methods include `GetLastErrorSeverity`, which responds with the severity of the error, and `GetLastStatus`, which responds with the text of the last status message returned by the `OnShowStatus` callback.

Also included are two user-defined error methods: `GetLastErrorUserCode` and `GetLastErrorUserString`. The `GetLastErrorUserCode` returns a user-defined error code, and `GetLastErrorUserString` returns a user-defined string that describes the last error that occurred. If you have designed a custom plug-in for RealPlayer, you can use both of these methods to return your own error codes and error strings for your plug-in. If you are not providing a custom plug-in for RealPlayer, both of these methods return nothing.

Seeking Through a Clip

.....

`OnPreSeek`	`OnPostSeek`

The Embedded RealPlayer provides several methods that can be used to obtain information about the currently-playing clip, including determining the present clip position, total length, and whether you can seek through the presentation.

During playback, you can use the `GetCanSeek` method to verify whether you can seek through the clip. When the method returns `true`, you can then use the

`GetLength` and `GetPosition` methods to obtain the clip length and current position in the clip. The actual length is returned as the total number of milliseconds in the clip. The position is returned as the number of milliseconds already played. You can also use the information returned from the `GetLength` method to ensure that the postion entered in the `SetPosition` method does not exceed the total length of the clip.

Another method that can be used when the `GetCanSeek` method returns `true` is `SetCanSeek`. This method allows you to specify whether a clip can be seeked. However, if you use the `SetCanSeek` method to allow seeking through an inherently unseekable clip, such as a live broadcast, the set method will have no effect.

APPENDIX A

METHODS

An application, applet, or control can use the methods described in this chapter to communicate with RealPlayer. The methods are listed here in alphabetical order and each description contains information about how to use the method in your Netscpae plug-in or ActiveX control, an example of syntax and usage, and backward compatibilty tips for various API versions.

For More Information: For information about categories of methods, such as those used to control playback of your presentation, see Chapter 2 "Using Methods and Callbacks".

Many of the methods listed below return a `boolean` value for plug-ins. Plug-in developers can safely ignore the `boolean` value since it should always return `true`. This value will only be `false` if some serious error occurs with the plug-in. Therefore, if you are having problems getting your script to work, you might want to check this return value.

AboutBox

COMMENT:	**OBSOLETE**	**AboutBox**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**SetShowAbout**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **ActiveX OnlySetShowAbout** in lieu of this function.

CanPlayPause

COMMENT:	**OBSOLETE**	**CanPlayPause**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**CanPause** and **CanPlay**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **CanPause** and **CanPlay** in lieu of this function.

CanPause

COMMENT:	**REPLACEMENT FOR**	**CanPlayPause**	Netscape and ActiveX

Indicates whether the player is currently playing a clip that can be paused.

```
CanPause(void)
```

Returns `true` if the player is currently playing a clip. Returns `false` if the player is already pause or is stopped.

CanPlay

COMMENT:	**REPLACEMENT FOR**	**CanPlayPause**	Netscape and ActiveX

Indicates whether the player is currently paused or stopped, or is currently playing.

```
CanPlay(void)
```

Returns `true` if the player is currently paused or stopped, and current source file is valid. Returns `false` if the player is currently playing.

CanStop

Indicates whether the the current clip is playing or paused, or the clip is already stopped. This method is compatible with RealPlayer version 5.0 and later.

```
CanStop(void)
```

Returns `true` if RealPlayer is currently playing a clip or is paused. Returns `false` if the clip is already stopped.

DoGotoURL

Causes the control to attempt a navigation to the specified URL in the specified frame target. The container must support URL browsing. This method is backward-compatible with ActiveX controls built with RealPlayer version 5.0 or later, but not compatible with version 5.0 Netscape plug-ins.

```
DoGotoURL(string url, string target)
```

url
The URL to which to navigate.

target
The frame target to which to navigate.

The target parameter is required in order to use this function, but the value of the parameter is ignored.

Returns `void`.

DoNextItem

COMMENT:	**OBSOLETE**	**DoNextItem**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**DoNextEntry**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **DoNextEntry** in lieu of this function.

DoNextEntry

COMMENT:	**REPLACEMENT FOR**	**DoNextItem**	Netscape and ActiveX

Skips to the next clip in the RAM (`.ram` or `.rpm`) or SMIL file that contains multiple clips. In a SMIL file, a `<par>` group is treated as a single clip.

```
DoNextEntry(void)
```

Returns a `boolean` value for plug-ins.

DoPlayPause

COMMENT:	**OBSOLETE**	**DoPlayPause**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**DoPause** and **DoPlay**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **DoPause** and **DoPlay** in lieu of this function.

DoPause

COMMENT:	**REPLACEMENT FOR**	**DoPlayPause**	Netscape and ActiveX

Pauses the current clip. Equivalent to clicking the Pause button.

```
DoPause(void)
```

Returns a `boolean` value for plug-ins.

DoPlay

COMMENT: REPLACEMENT FOR DoPlayPause Netscape and ActiveX

Plays the current clip. Equivalent to clicking the Play button.

```
DoPlay(void)
```

Returns a `boolean` value for plug-ins.

DoPrevItem

COMMENT: OBSOLETE DoPrevItem Netscape and ActiveX

COMMENT: REPLACED BY DoPrevEntry Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **DoPrevEntry** in lieu of this function.

DoPrevEntry

COMMENT: REPLACEMENT FOR DoPrevItem Netscape and ActiveX

Skips to the previous clip in a RAM (`.ram` or `.rpm`) or SMIL file that contains multiple clips. In a SMIL file, a `<par>` group is treated as a single clip.

```
DoPrevEntry(void)
```

Returns a `boolean` value for plug-ins.

DoStop

Stops the clip. Equivalent to clicking the Stop button. This method is compatible with RealPlayer version 5.0 and later.

```
DoStop(void)
```

Returns a `boolean` value for plug-ins.

EditPreferences

COMMENT: OBSOLETE EditPreferences Netscape and ActiveX

COMMENT: REPLACED BY SetShowPreferences Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **SetShowPreferences** in lieu of this function.

GetAuthor

Indicates the current clip's author string.

`GetAuthor(void)`

Returns a `string` that contains the current clip's author.

GetAutoGoToURL

COMMENT: Method name differs between APIs - See the Warning description below

Indicates whether or not the **AutoGoToURL** setting is enabled.

`GetAutoGoToURL(void)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, the name of the method is **`GetAutoGotoURL`**, while in Java, the name is **`GetAutoGoToURL`**. Either name may be used when developing in JavaScript or VBScript.

Returns `true` if the setting is enabled. Returns `false` if the setting is disabled.

GetAutoStart

Indicates whether or not playback will start automatically.

`GetAutoStart(void)`

Returns `true` if playback will start automatically. Returns `false` if the playback will not be started automatically.

GetBackgroundColor

Indicates the hexadecimal value for the current background color for the image window.

`GetBackgroundColor(void)`

Returns a `string` that contains the RGB hexadecimal color value in the format `#RRGGBB`. The color names for these color values are described in `SetBackgroundColor` on page 67.

GetBandwidthAverage

Indicates the average amount of bandwidth used by the presentation.

```
GetBandwidthAverage(void)
```

Returns an `int32` that contains the average bandwidth, in bits per second, of the packet transfer from the beginning of the playback to the current time.

GetBandwidthCurrent

Indicates the current amount of bandwidth being used by the presentation.

```
GetBandwidthCurrent(void)
```

Returns an `int32` that contains the current bandwidth in bits per second.

GetBufferingTimeElapsed

Indicates the current elapsed buffering time.

```
GetBufferingTimeElapsed(void)
```

Returns an `int32` that contains the number of milliseconds of elapsed buffering time.

GetBufferingTimeRemaining

Indicates the estimated remaining buffering time.

```
GetBufferingTimeRemaining(void)
```

Returns an `int32` that contains the estimated remaining buffering time in milliseconds.

GetCanSeek

Indicates whether the user can seek within the clip through the user interface.

```
GetCanSeek(void)
```

Returns `true` if the user can seek within the clip. Returns `false` if the user cannot seek within the clip. Live or simulated live clips always return `false`.

GetCenter

Indicates whether or not the visual datatype will be centered within the image window.

```
GetCenter(void)
```

Returns `true` if the visual datatype is centered in the image window. Returns `false` (default) if the datatype is not centered.

GetClipHeight

Indicates the height of the presentation.

```
GetClipHeight(void)
```

Returns an `int32` that contains the height of the clip window, in pixels. A value of 0 is returned when the presentation is not visual.

GetClipWidth

Indicates the width of the presentation.

```
GetClipWidth(void)
```

Returns an `int32` that contains the width of the clip window, in pixels. A value of 0 is returned when the presentation is not visual.

GetConnectionBandwidth

Indicates the normal, maximum bandwidth settings as set by the user in the RealPlayer preferences.

```
GetConnectionBandwidth(void)
```

Returns an `int32` that contains the maximum bandwidth setting, in bits per second, from the Connections tab of the RealPlayer Preferences dialog.

GetConsole

Indicates a console name used to link multiple control instances.

`GetConsole(void)`

Returns a `string` that contains the name of the RealPlayer console currently associated with the embedded control.

GetConsoleEvents

Indicates whether console events are enabled.

`GetConsoleEvents(void)`

Returns `true` if console events are enabled. Returns `false` if console events are disabled.

For More Information: See "Event Handling" on page 31 for an explanation of console events.

GetControls

Indicates the name of the visible components of the RealPlayer control.

`GetControls(void)`

Returns a `string` that contains the name of the RealPlayer control currently associated with the name or ID of the embedded control.

For More Information: For valid control names, see the chapter on Web page playback in *RealSystem G2 Production Guide* at **http://service.real.com/help/library/encoders.html**.

GetCopyright

Indicates the current clip's copyright string.

`GetCopyright(void)`

Returns a `string` that contains the current clip's copyright information.

GetCurrentEntry

COMMENT: Return type differs between APIs - See the Warning description below

Indicates the number of the entry currently playing.

`GetCurrentEntry(void)`

Returns an `int32` that contains the number of the entry currently playing. The current entry number of the first entry is "0".

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function returns an `int16`, while in Java, an `int32` value is returned. Either integer type may be used when developing in JavaScript or VBScript.

string GetDRMInfo

This method is used in conjunction with the RealNetworks digital rights management systems. The returned string provides necessary client information that is used by the license server to generate content licenses for a particular unique user.

```
GetDRMInfo (4-letter product identifier)
```

```
Returns a string.
```

```
where the product acronym is:
```

```
RNBA for use with the Media Commerce Suite
```

The returned string is in the following form (line breaks are for readability only):

```
ClientPubKey=<ClientPubKey>
&Challenge=<Challenge>
&ExtraInfo=<ExtraInfo>
```

> **Note:** This method is available only in Embedded RealPlayer builds 6.0.8.1024 and later.

GetDoubleSize

Indicates whether or not the image is currently in double-size mode.

```
GetDoubleSize(void)
```

Returns `true` if the image is double size. Returns `false` if the image is not double size.

> **Note:** This method is included only for ActiveX controls and plug-ins used in applications; this method is not intended for use in web pages.

GetEnableContextMenu

Indicates whether or not the context menu is currently enabled.

`GetEnableContextMenu(void)`

Returns `true` (default) if the context menu is enabled. Returns `false` if the context menu is disabled.

GetEnableDoubleSize

Indicates whether or not the double size selection is enabled in the Zoom item of the context menu.

`GetEnableDoubleSize(void)`

Returns `true` if the double size selection is enabled. Returns `false` (default) if the double size selection is disabled.

GetEnableFullScreen

Indicates whether or not the full screen selection is enabled in Zoom item of the context menu.

`GetEnableFullScreen(void)`

Returns `true` (default) if the full screen selection is enabled. Returns `false` if the full screen selection is disabled.

COMMENT: Netscape and ActiveX**Original Description for GetEnableMessageBox**

COMMENT: **Indicates whether error message dialogs can be displayed.**

COMMENT: **GetEnableMessageBox(void)**

COMMENT: **Returns** `true` **if error message dialogs are enabled. Returns** `false` **if error message dialogs are not displayed.**

GetEnableOriginalSize

Indicates whether the original size selection is enabled in the Zoom item of the context menu.

`GetEnableOriginalSize(void)`

Returns `true` (default) if the original size option is enabled. Returns `false` if the original size option is disabled.

GetEntryAbstract

COMMENT: Parameter type differs between APIs - See the Warning description below

Indicates the abstract for the specified playlist entry.

`GetEntryAbstract(int32 entry_index)`

entry_index
The entry number of the clip in the playlist for which the abstract is being requested. The entry number for the first clip in the playlist is “0”.

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns a `string` that contains the abstract for the specified playlist entry.

GetEntryAuthor

COMMENT: Parameter type differs between APIs - See the Warning description below

Indicates the author for the specified playlist entry.

`GetEntryAuthor(int32 entry_index)`

entry_index
The entry number of the clip in the playlist for which the author is being requested. The entry number for the first clip in the playlist is “0”.

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns a `string` that contains the author for the specified playlist entry.

GetEntryCopyright

COMMENT: Parameter type differs between APIs - See the Warning description below

Indicates the copyright for the specified playlist entry.

`GetEntryCopyright(int32 entry_index)`

entry_index
The entry number of the clip in the playlist for which the copyright is being requested. The entry number for the first clip in the playlist is "0".

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns a `string` that contains the copyright for the specified playlist entry.

GetEntryTitle

COMMENT: Parameter type differs between APIs - See the Warning description below

Indicates the title for the specified playlist entry.

```
GetEntryTitle(int32 entry_index)
```

entry_index
The entry number of the clip in the playlist for which the title is being requested. The entry number for the first clip in the playlist is "0".

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns a `string` that contains the title for the specified playlist entry.

GetFullScreen

Indicates whether or not the image is currently in full-screen mode.

```
GetFullScreen(void)
```

Returns `true` if the image is in full-screen mode. Returns `false` if the image is not in full-screen mode.

GetImageStatus

Indicates whether the status text is written to the image window.

`GetImageStatus(void)`

Returns `true` (default) if the status text is written to the image window. Returns `false` if the status text is not sent.

GetIsPlus

Indicates whether the client is a basic RealPlayer or RealPlayer Plus.

`GetIsPlus(void)`

Returns `true` if the client is RealPlayer Plus. Returns `false` if the client is the basic RealPlayer.

GetLastErrorMoreInfoURL

Provides the "more info" URL from the last error.

`GetLastErrorMoreInfoURL(void)`

Returns a `string` that contains the "more info" URL. This method may return nothing (for example, if there is no "more info" URL).

GetLastErrorRMACode

Gets the RMA error code from the last error. RMA error codes are described in the header file `pnresult.h` in the RealSystem G2 SDK available at **http://www.realnetworks.com/devzone/**. In normal operation, all RealSystem components need to be able to handle the following basic codes that may be returned by the RealSystem system:

- PNR_FAIL — Operation failed.
- PNR_OK — Operation succeeded.
- PNR_UNEXPECTED — Call was unexpected or method is not implemented.

`GetLastErrorRMACode(void)`

Returns an `int32` that contains the error code value.

COMMENT: Netscape and ActiveX**Original Description for GetLastErrorRMACodeString**

COMMENT: **Gets the RMA error code string from the last error.**

COMMENT: **This method is intended for a Netscape control only.**

COMMENT: **GetLastErrorRMAString(void)**

COMMENT: Returns a `string` that contains a text description of the last error code.

GetLastErrorSeverity

COMMENT: Return type differs between APIs - See the Warning description below

Indicates the error level for the last error.

`GetLastErrorSeverity(void)`

Returns an `int32` that contains the error level.

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function returns an `int16`, while in Java, an `int32` value is returned. Either integer type may be used when developing in JavaScript or VBScript.

Error levels consist of the following:

Error Levels

Level	Condition	Usage
0	Panic	Error potentially causing a system failure. RealSystem takes actions necessary to correct the problem. This may include shutting down the presentation.
1	Severe	Error requiring immediate user intervention to prevent a problem. RealSystem will shut down the presentation if necessary.
2	Critical	Error that may require user intervention to correct. RealSystem will shut down the presentation if necessary.
3	General	Error that does not cause a significant problem with normal system operation.
4	Warning	Warning about a condition that does not cause system problems but may require attention.
5	Notice	Notice about a condition that does not cause system problems but should be noted.
6	Informational	Informational message only.
7	Debug	Information of use only when debugging a program.

GetLastErrorUserCode

Indicates the user error code from the last error.

```
GetLastErrorUserCode(void)
```

Returns an `int32` that contains the user error code. This method will always return 0 unless you are using a custom plug-in that provides its own user-defined error codes for error events.

GetLastErrorUserString

Gets the error string from the last error dialog.

```
GetLastErrorUserString(void)
```

Returns a `string` that contains the last error message. This method will return nothing unless you are using a custom plug-in that provides its own user-defined error strings for error events.

Netscape OnlyGetLastMessage

COMMENT: Identical to GetLastStatus - See the Note description below

Gets the text of the last status message that was returned by the `OnShowStatus` callback method.

> **Note:** This method is intended for a Netscape plug-in only. If you are coding an ActiveX control, use the **Netscape OnlyGetLastMessage** method instead.

```
GetLastMessage(void)
```

Returns a `string` that contains the last status message.

GetLastStatus

COMMENT: Identical to GetLastMessage - See the Note description below

Gets the text of the last status message that was returned by the `OnShowStatus` callback method.

> **Note:** This method is intended for an ActiveX control only. If you are coding a Netscape plug-in, use the GetLastStatus method instead.

```
GetLastStatus(void)
```

Returns a `string` that contains the last status message.

GetLength

Indicates the total length of the clip.

`GetLength(void)`

Returns an `int32` that contains the total length of the clip, in milliseconds. Valid values are `>=0`.

GetLiveState

Indicates whether the current clip is live.

`GetLiveState(void)`

Returns `true` if the current clip is live. Returns `false` if the current clip is not live.

GetLoop

Indicates whether the clip has been set to loop.

`GetLoop(void)`

Returns `true` if the clip has been set to loop until play is interrupted. Returns `false` if the clip does not loop (default).

GetMaintainAspect

Indicates whether or not the aspect ratio of the visual datatype will be maintained.

`GetMaintainAspect(void)`

Returns `true` if the aspect ratio of the visual datatype is maintained. Returns `false` (default) if the aspect ratio changes when the image window is stretched.

GetMute

Indicates whether or not the volume has been muted.

`GetMute(void)`

Returns `true` if the volume is muted. Returns `false` if the volume is not muted.

GetNoLabels

COMMENT: OBSOLETE GetNoLabels Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later.

COMMENT: Original Description for GetNoLabels

COMMENT: Indicates whether or not the title, author, and copyright labels will be suppressed.

COMMENT: GetNoLabels(void)

COMMENT: Returns `true` **if the title, author, and copyright labels are suppressed. Returns** `false` **if these labels are displayed.**

GetNoLogo

Indicates whether or not the Real logo will be displayed in the image window.

`GetNoLogo(void)`

Returns `true` if the Real logo is not displayed in the image window. Returns `false` (default) if the Real logo is displayed in the image window.

GetNumEntries

COMMENT: Return type differs between APIs - See the Warning description below

Indicates the total number of entries in the playlist.

`GetNumEntries(void)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function returns an `int16`, while in Java, an `int32` value is returned. Either integer type may be used when developing in JavaScript or VBScript.

Returns an `int32` that contains the total number of entries in the playlist. The entry number for a single entry is "1".

GetNumLoop

Indicates the number of times the clip is set to loop.

`GetNumLoop(void)`

Returns an `int32` that indicates the number of times the clip has been set to loop by `SetNumLoop`.

GetNumSources

COMMENT: Return type differs between APIs - See the Warning description below

Indicates the number of sources in the presentation.

`GetNumSources(void)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function returns an `int16`, while in Java, an `int32` value is returned. Either integer type may be used when developing in JavaScript or VBScript.

Returns an `int32` that contains the number of sources in the presentation.

GetOriginalSize

Indicates whether the image is currently in its original size.

`GetOriginalSize(void)`

Returns `true` if the image is its original size. Returns `false` if the image is not its original size.

GetPacketsEarly

Returns the total number of packets received from the RealSystem Server before they are ready to play.

> **Note:** This method is intended for an ActiveX control only.

`GetPacketsEarly(void)`

Returns an `int32` that contains the number of packets that were received too early.

GetPacketsLate

Indicates the total number of packets received from the RealSystem Server that are too late to play.

`GetPacketsLate(void)`

Returns an `int32` that contains the number of packets that were received too late.

GetPacketsMissing

Indicates the total number of packets not received from the RealSystem Server in time to play.

`GetPacketsMissing(void)`

Returns an `int32` that contains the number of packets that were not received in time to play.

GetPacketsOutOfOrder

Indicates the total number of packets received from the RealSystem Server out of order.

`GetPacketsOutOfOrder(void)`

Returns an `int32` that contains the total number of out of order packets.

GetPacketsReceived

Indicates the total number of packets that have currently been received from the RealSystem Server.

`GetPacketsReceived(void)`

Returns an `int32` that contains the total number of packets received so far.

GetPacketsTotal

Indicates the total number of packets currently used by the presentation. The total number of packets reported by this method include the number of received packets plus the number of lost packets. If there are no lost packets, this method returns the same number as `GetPacketsReceived`.

`GetPacketsTotal(void)`

Returns an `int32` that contains the total number of packets.

GetPlayState

Indicates the current state of the RealPlayer.

`GetPlayState(void)`

Returns an `int32` value with the following meanings:

- 0 — Stopped
- 1 — Contacting
- 2 — Buffering
- 3 — Playing
- 4 — Paused
- 5 — Seeking

GetPosition

Indicates the current position in the clip.

`GetPosition(void)`

Returns an `int32` that contains the current position in the clip, in milliseconds. Valid values are `>=0` and `<=`*`total clip length`*.

GetPreferredLanguageID

Indicates the preferred language ID.

`GetPreferredLanguageID(void)`

Returns an `int32` that contains the identification number of the language set in the Content tab of the RealPlayer Preferences dialog.

COMMENT: Research language IDs and insert findings here and at GetUserCountryID

GetPreferredLanguageString

Indicates the preferred language for content as set by the user in the RealPlayer preferences.

`GetPreferedLanguageString(void)`

Returns a `string` that contains the preferred language. For a list of language codes, see the SMIL language codes appendix in *RealSystem G2 Production Guide*.

GetPreFetch

Indicates whether or not `PREFETCH` is enabled.

`GetPreFetch(void)`

Returns `true` if `PREFETCH` is enabled. Returns `false` if `PREFETCH` is not enabled.

ActiveX Only GetShowAbout

Indicates whether or not the About box is open.

```
GetShowAbout(void)
```

Returns `true` if the About dialog box is visible. Returns `false` if the dialog box is not visible.

GetShowPreferences

Indicates whether or not the Preferences dialog box is visible.

```
GetShowPreferences(void)
```

Returns `true` if the Preferences dialog box is visible. Returns `false` if the dialog box is not visible.

GetShowStatistics

COMMENT: REPLACEMENT FOR **IsStatisticsVisible** Netscape and ActiveX

Indicates whether or not the RealPlayer Statistics dialog box is visible.

```
GetShowStatistics(void)
```

Returns `true` if the RealPlayer Statistics dialog box is visible. Returns `false` (default) if the dialog box is not visible.

GetShuffle

Indicates whether or not shuffle play is enabled.

```
GetShuffle(void)
```

Returns `true` if shuffle play is enabled. Returns `false` if shuffle play is disabled.

GetSource

Indicates the URL of the playing clip.

```
GetSource(void)
```

Returns a `string` that contains the URL of the playing clip.

GetSourceTransport

COMMENT: Parameter type differs between APIs - See the Warning description below

Returns a string with the source protocol used for playback.

`GetSourceTransport(int32 source_number)`

source_number

The number of the source for which a protocol will be specified. This number can be set between 1 and *n*, where *n* is the number of sources returned by `GetNumSources`.

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns a `string` that contains the source protocol used for playback, such as UDP or TCP.

GetStereoState

Indicates whether the current clip is in stereo.

`GetStereoState(void)`

Returns `true` if the current clip is in stereo and `false` for monaural. This function returns a `boolean` value for Netscape plug-ins

GetTitle

Indicates the current clip's title string.

`GetTitle(void)`

Returns a `string` that contains the current clip's title.

GetUserCountryID

Indicates the country that the user selected during electronic registration.

`GetUserCountryID(void)`

Returns an `int32` that contains the identification number of the user-selected country.

COMMENT: Research language IDs and insert findings from GetPreferredLanguageID

GetVersionInfo

Indicates major and minor version information for the Embedded RealPlayer (not the parent RealPlayer).

`GetVersionInfo(void)`

Returns a `string`, such as 6.0.0.128, that contains the version information.

Netscape Only GetVolume

COMMENT: Return type differs between APIs - See the Warning description below

Indicates the current volume level.

`GetVolume(void)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function returns an `int16`, while in Java, an `int32` value is returned. Either integer type may be used when developing in JavaScript or VBScript.

Returns an `int32` that contains the current volume level. The returned value will be in the range of 0 through 100.

GetWantErrors

Indicates whether error dialogs will be displayed.

`GetWantErrors(void)`

Returns `true` if error dialogs are trapped, and therefore not displayed. Returns `false` if the error dialogs are displayed.

GetWantKeyboardEvents

Indicates whether keyboard events are sent or not (that is, it indicates whether the `OnKeyDown`, `OnKeyPress`, and `OnKeyUp` callbacks are to be sent).

`GetWantKeyboardEvents(void)`

Returns `true` if the keyboard events are sent. Returns `false` (default) if the keyboard events are not sent.

GetWantMouseEvents

Indicates whether or not mouse events are to be sent (that is, whether the `OnLButtonDown`, `OnLButtonUp`, `OnMouseMove`, `OnRButtonDown`, and `OnRButtonUp` callbacks are to be sent).

```
GetWantMouseEvents(void)
```

Returns `true` if the mouse events are sent. Returns `false` (default) if the mouse events are ignored.

HasNextItem

COMMENT:	**OBSOLETE**	**HasNextItem**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**HasNextEntry**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **HasNextEntry** in lieu of this function.

HasNextEntry

COMMENT:	**REPLACEMENT FOR**	**HasNextItem**	Netscape and ActiveX

Tests if the next clip function is available. The next clip function is available when the connected source is a RAM (`.ram` or `.rpm`) or SMIL file that contains multiple clips and the current clip is not the last clip in the RAM or SMIL file. In a SMIL file, a `<par>` group is treated as a single clip.

```
HasNextEntry(void)
```

Returns `true` if the next clip function is available. Returns `false` if no more clips are available after the current clip.

HasPrevItem

COMMENT:	**OBSOLETE**	**HasPrevItem**	Netscape and ActiveX
COMMENT:	**REPLACED BY**	**HasPrevEntry**	Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **HasPrevEntry** in lieu of this function.

HasPrevEntry

COMMENT: **REPLACEMENT FOR** **HasPrevItem** Netscape and ActiveX

Tests if the previous clip function is available. The previous clip function is available when the connected source is a RAM (`.ram` or `.rpm`) or SMIL file that contains multiple clips and the current clip is not the first clip in the RAM file. In a SMIL file, a `<par>` group is treated as a single clip.

```
HasPrevEntry(void)
```

Returns `true` if the previous clip function is available. Returns `false` if the current clip is the first clip.

HideShowStatistics

COMMENT: **OBSOLETE** **HideShowStatistics** Netscape and ActiveX

COMMENT: **REPLACED BY** **SetShowStatistics** Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **SetShowStatistics** in lieu of this function.

Netscape OnlyIsStatisticsVisible

COMMENT: **OBSOLETE** **IsStatisticsVisible** Netscape and ActiveX

COMMENT: **REPLACED BY** **GetShowStatistics** Netscape and ActiveX

Although this method is obsolete, backward-compatibility is supported for RealPlayer versions 5.0 and later. It is strongly recommended that you use **GetShowStatistics** in lieu of this function.

Netscape OnlyNetscape OnlySetAuthor

Sets the current clip's author string, overriding any existing author information. `GetAuthor` subsequently returns this new value.

```
SetAuthor(string new_author)
```

new_author
: The author string to be set. This author string overrides all subsequent author information in a multiclip presentation.

Returns a `boolean` value for plug-ins.

SetAutoGoToURL

COMMENT: Method name differs between APIs - See the Warning description below

Specifies how a URL will be handled. This method is compatible with RealPlayer version 5.0 and later.

`SetAutoGoToURL(boolean enable_start)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, the name of the method is **`SetAutoGotoURL`**, while in Java, the name is **`SetAutoGoToURL`**. Either name may be used when developing in JavaScript or VBScript.

enable_start
: If set to `true`, a RealPlay plug-in automatically forwards the URL event to the browser. If set to `false`, the `onGoToURL` event is handled by a Java applet or VBScript instead.

> **For More Information:** Beginning with RealPlayer G2, this can also be set with the `AUTOGOTOURL` parameter in the `<EMBED>` or `<OBJECT>` tag.

Returns a `boolean` value for plug-ins.

SetAutoStart

COMMENT: OBSOLETE **SetAutoStart** Netscape

Sets whether or not the control automatically starts playing once the source data is available. This method is backward-compatible with Netscape plug-ins and ActiveX controls built with RealPlayer version 5.0 or later.

> **For More Information:** If you are developing a Netscape plug-in in RealPlayer version 5.0 or later, you can also use the `AUTOSTART` parameter defined on page 96, to specify automatic playback in the tag definition.

```
SetAutoStart(boolean auto_start)
```

auto_start

If set to `true`, the control automatically starts playing once the source data is available. If set to `false`, the control does not automatically start playing.

Returns a `boolean` value for plug-ins.

SetBackgroundColor

Specifies the desired background color for the image window control.

> **For More Information:** You can also use the `BACKGROUNDCOLOR` parameter defined on page 97, to specify the background color of the image window in the tag definition.

```
SetBackgroundColor(string color)
```

color

The background color of the image window control. Valid values are an RGB hexadecimal color value in the format `#RRGGBB`, or the following color names, shown here with their corresponding RGB values:

<TD bgcolor=#FFFFFF><FONT COLOR="#000000">white (#FFFFFF)</FONT></TD>	<TD bgcolor=#C0C0C0><FONT COLOR="#000000">silver (#C0C0C0)</FONT></TD>	<TD bgcolor=#808080><FONT COLOR="#FFFFFF">gray (#808080)</FONT></TD>	<TD bgcolor=#000000><FONT COLOR="#FFFFFF">black (#000000)</FONT></TD>
<TD bgcolor=#FFFF00><FONT COLOR="#000000">yellow (#FFFF00)</FONT></TD>	<TD bgcolor=#FF00FF><FONT COLOR="#FFFFFF">fuchsia (#FF00FF)</FONT></TD>	<TD bgcolor=#FF0000><FONT COLOR="#FFFFFF">red (#FF0000)</FONT></TD>	<TD bgcolor=#800000><FONT COLOR="#FFFFFF">maroon (#800000)</FONT></TD>
<TD bgcolor=#00FF00><FONT COLOR="000000">lime (#00FF00)</FONT></TD>	<TD bgcolor=#808000><FONT COLOR="#FFFFFF">olive (#808000)</FONT></TD>	<TD bgcolor=#008000><FONT COLOR="#FFFFFF">green (#008000)</FONT></TD>	<TD bgcolor=#800080><FONT COLOR="#FFFFFF">purple (#800080)</FONT></TD>

<TD bgcolor=#00FFFF> <FONT COLOR="#000000">aqua (#00FFFF)</FONT> </TD>	<TD bgcolor=#008080> <FONT COLOR="#FFFFFF">teal (#008080)</FONT></TD>	<TD bgcolor=#0000FF><FONT COLOR="#FFFFFF">blue (#0000FF)</FONT></TD>	<TD bgcolor=#000080> <FONT COLOR="#FFFFFF">navy (#000080)</FONT></TD>

Returns a `boolean` value for plug-ins.

SetCanSeek

Sets whether the user can seek within the clip through the user interface.

```
SetCanSeek(boolean can_seek)
```

can_seek
If set to `true` (default), the user can seek within the clip. If set to `false`, the user cannot seek within the clip. This function cannot be used to establish seeking ability for a live or simulated live clip.

Returns a `boolean` value for plug-ins.

SetCenter

Sets whether or not the visual datatype should be centered at its natural size within the image window.

For More Information: You can also use the `CENTER` parameter defined on page 98, to specify that the presentation should be centered in the image window, in the tag definition.

```
SetCenter(boolean value)
```

value
If set to `true`, the visual datatype is centered in the image window at its natural size. If set to `false` (default), the visual datatype's height and width is expanded to fill the image window.

Note: The `SetCenter` and `SetMaintainAspect` methods cannot both be set to `true`. Therefore, if you have set the `set` parameter of the `SetMaintainAspect` method to `true`, the `value` parameter of the `SetCenter` method must be set to `false`.

Returns a `boolean` value for plug-ins.

SetConsoleName

COMMENT:	**OBSOLETE**	**SetConsoleName**	Netscape Only
COMMENT:	**REPLACED BY**	**SetConsole**	Netscape and ActiveX

This method is backward-compatible with Netscape plug-ins built with RealPlayer version 5.0 or later, but not compatible with version 5.0 ActiveX controls. It is strongly recommended that you use SetConsole in lieu of this function.

SetConsole

COMMENT:	**REPLACEMENT FOR**	**SetConsoleName**	Netscape

Sets a console name used to link multiple control instances. Call this once for each instance of a control you want to link. All controls with the same console name work together. For example, if you have multiple Play and Stop buttons on the same page, a shared console name enables them to control the same clip. The console name `_master` links to all instances. The console name `_unique` links to no other instances.

> **For More Information:** You can also use the `CONSOLE` parameter defined on page 100, to specify whether your controls are linked in the tag definition. For more information about linking multiple controls, see the *RealSystem Production Guide*.

```
SetConsole(string console)
```

console

The name of the console to be set. This name must be associated with the unique name or ID for each embedded control you want to link. For example:

`document.playcontrol.SetConsole("console1")` — JavaScript

`Document.playcontrol.SetConsole("console1")` — VBScript

Returns a `boolean` value for plug-ins.

SetConsoleEvents

Sets whether or not console events are enabled. This method is included to provide more control of callbacks in plug-ins.

This method does not affect ActiveX controls.

`SetConsoleEvents(boolean value)`

value
If set to `true`, events from any plug-in are sent to the applet. If set to `false`, only the plug-in to which the console connects will send events.

Returns a `boolean` value for plug-ins.

For More Information: See "Event Handling" on page 31 for an explanation of console events.

Netscape OnlyNetscape Only

SetControlString

COMMENT: **OBSOLETE** **SetControlString** Netscape

COMMENT: **REPLACED BY** **SetControls** Netscape and ActiveX

This method is backward-compatible with Netscape plug-ins built with RealPlayer version 5.0 or later, but not compatible with version 5.0 ActiveX controls.. It is strongly recommended that you use SetControls in lieu of this function.

SetControls

COMMENT: **REPLACEMENT FOR** **SetControlString** Netscape

Sets the visible components of the control.

For More Information: You can also use the `CONTROLS` parameter defined on page 100, to add controls to your Web page in the tag definition.

`SetControls(string controls)`

controls
The name of the RealPlayer control to be set. This name must be associated with a unique name or ID for each embedded control. For example:

`document.playcontrol.SetControls("PlayOnlyButton")` — JavaScript

`Document.playcontrol.SetControls("PlayOnlyButton")` — VBScript

Returns a `boolean` value for plug-ins.

SetCopyright

Sets the current clip's copyright string, overriding any existing copyright information. `GetCopyright` subsequently returns this new value.

```
SetCopyright(string copyright)
```

copyright
The copyright string to be set. This copyright string overrides all subsequent copyright information in a multiclip presentation.

Returns a `boolean` value for plug-ins.

SetDoubleSize

Sets the image window to double its original size.

```
SetDoubleSize(void)
```

Returns a `boolean` value for plug-ins.

> **Note:** This method is included only for ActiveX controls and plug-ins used in applications; this method is not intended for use in Web pages.

SetEnableContextMenu

Specifies whether the control over which the mouse cursor is placed displays the default context menu when the right mouse button is pressed.

```
SetEnableContextMenu(boolean menu_on)
```

menu_on
If set to `true` (default), the context menu is displayed. If set to `false`, the context menu is not displayed.

Returns a `boolean` value for plug-ins.

SetEnableDoubleSize

Enables or disables the double size selection in the Zoom item of the context menu.

```
SetEnableDoubleSize(boolean enabled)
```

enabled
If set to `true`, the double size selection is enabled. If set to `false` (default), the double size selection is disabled.

Returns a `boolean` value for plug-ins.

SetEnableFullScreen

Enables or disables the full screen selection in the Zoom item of the context menu.

```
SetEnableFullScreen(boolean enabled)
```

enabled
If set to `true` (default), the full screen selection is enabled. If set to `false`, the full screen selection is disabled.

Returns a `boolean` value for plug-ins.

SetEnableOriginalSize

Enables or disables the original size selection in the Zoom item of the context menu.

```
SetEnableOriginalSize(boolean enabled)
```

enabled
If set to `true` (default), the original size selection is enabled. If set to `false`, the original size selection is disabled.

Returns a `boolean` value for plug-ins.

SetFullScreen

Sets the image to full-screen mode.

```
SetFullScreen(void)
```

Returns a `boolean` value for plug-ins.

Note: The user presses the**Esc** key to reduce the image back to its original size.

SetImageStatus

Enables or disables the status text that is written along the bottom of the image window.

For More Information: You can also use the `IMAGESTATUS` parameter defined on page 102, to specify whether the status information should be displayed, in the tag definition.

```
SetImageStatus(boolean enabled)
```

enabled
If set to `true` (default), the status text is written to the image window. If set to `false`, the status text is not sent to the image window.

Returns a `boolean` value for plug-ins.

SetLoop

Specifies whether the clip will loop or not.

For More Information: You can also use the `LOOP` parameter defined on page 103, to specify whether the clip should loop, in the tag definition.

```
SetLoop(boolean set)
```

set
If set to `true`, the clip loops until play is interrupted. If set to `false`, the clip does not loop (default).

Returns a `boolean` value for plug-ins.

SetMaintainAspect

Sets whether or not to maintain the correct aspect ratio of the source within the image window when the image window is stretched.

For More Information: You can also use the `MAINTAINASPECT` parameter defined on page 103, to specify whether the correct aspect ratio should be maintained, in the tag definition.

```
SetMaintainAspect(boolean set)
```

set
If set to `true`, the correct aspect ratio of the source is maintained. If set to `false` (default), the aspect ratio is changed so the source fills the image window.

Note: The `SetMaintainAspect` and `SetCenter` methods cannot both be set to `true`. Therefore, if you have set the `value` parameter of the `SetCenter` method to `true`, the `set` parameter of the `SetMaintainAspect` method must be set to `false`.

Returns a `boolean` value for plug-ins.

SetMute

Sets the mute state.

```
SetMute(boolean mute)
```

mute
If set to `true`, the audio is muted. If set to `false`, the sound is not muted.

Returns a `boolean` value for plug-ins.

SetNoLabels

COMMENT: **OBSOLETE** **SetNoLabels** Netscape

Sets whether to suppress the title, author, and copyright label text in the controls window of RealPlayer 5.0. The content text strings are still displayed. This method is backward-compatible with Netscape plug-ins built with RealPlayer version 5.0 or later, but not compatible with version 5.0 ActiveX controls.

For More Information: You can also use the `NOLABELS` parameter defined on page 105, to specify whether the presentation information should be displayed, in the tag definition.

```
SetNoLabels(boolean set)
```

set
If set to `true`, the title, author, and copyright label text is suppressed in the controls window. If set to `false`, the title, author, and copyright label text is displayed.

Returns a `boolean` value for plug-ins.

SetNoLogo

Sets whether to suppress the display of the Real logo in the image window control.

> **For More Information:** You can also use the `NOLOGO` parameter defined on page 106, to specify whether the logo should be displayed, in the tag definition.

```
SetNoLogo(boolean set)
```

set
If set to `true`, the Real logo is not displayed in the image window. The image window then defaults to black unless a background color has been specified. If set to `false` (default), the Real logo is displayed in the image window.

Returns a `boolean` value for plug-ins.

SetNumLoop

Sets number of times to loop the clip.

> **For More Information:** You can also use the `NUMLOOP` parameter defined on page 107, to specify the number of times the presentation should loop, in the tag definition.

```
SetNumLoop(int32 number_of_loops)
```

number_of_loops
The number of times for the clip to loop.

Returns a `boolean` value for plug-ins.

SetOriginalSize

Sets the image window to its original size.

```
SetOriginalSize(void)
```

Returns a `boolean` value for plug-ins.

SetPosition

Seeks into the clip to the specified point.

```
SetPosition(int32 position)
```

position
The point in the clip to which to seek, in milliseconds. Valid values are `>=0` through `<=`*`total clip length`*. If an attempt is made to set the position `>total length`, then `SetPosition` will equal `total length`.

> **Note:** Be sure to wait for the seek to finish before continuing with any programming that requires the clip to be at the point specified by this method.

Returns a `boolean` value for plug-ins.

SetPreFetch

Enables or disables `PREFETCH` playback mode.

```
SetPreFetch(boolean set)
```

set
If set to `true`, `PREFETCH` playback mode is enabled. If set to `false` (default), `PREFETCH` playback mode is disabled.

> **For More Information:** You can also use the `PREFETCH` parameter defined on page 108, to specify whether prefetch playback mode is enabled, in the tag definition.

Returns a `boolean` value for plug-ins.

ActiveX OnlySetShowAbout

COMMENT: REPLACEMENT FOR **AboutBox** Netscape and ActiveX

Displays the RealPlayer About dialog box.

```
SetShowAbout(boolean set)
```

set

If set to `true`, the RealPlayer About dialog box is displayed. Setting this parameter to `false` while the dialog box is displayed does nothing; you must close the display using the buttons available in the dialog box.

Returns a `boolean` value for plug-ins.

SetShowPreferences

COMMENT: REPLACEMENT FOR EditPreferences Netscape and ActiveX

Displays the RealPlayer Preferences dialog box.

```
SetShowPreferences(boolean set)
```

set

If set to `true`, the RealPlayer Preferences dialog box is displayed. Setting this parameter to `false` while the dialog box is displayed does nothing; you must close the display using the buttons available in the dialog box.

Returns a `boolean` value for plug-ins.

SetShowStatistics

COMMENT: REPLACEMENT FOR HideShowStatistics Netscape and ActiveX

Sets the RealPlayer Statistics dialog box to visible.

```
SetShowStatistics(boolean set)
```

set

If set to `true`, the RealPlayer Statistics dialog box is displayed. If set to `false` and the RealPlayer Statistics dialog box is visible, the dialog box will be closed.

Returns a `boolean` value for plug-ins.

SetShuffle

Randomizes playback of all clips, excluding clips that have already played. Works for multiclip RAM files (`.ram` or `.rpm`) or SMIL files that contain only a sequence of clips.

For More Information: You can also use the `SHUFFLE` parameter defined on page 110, to specify whether clip playback should be randomized, in the tag definition.

```
SetShuffle(boolean set)
```

set
If set to `true`, clip playback is randomized. If set to `false`, the clips are played back in the order in which they appear in the multiclip RAM file or SMIL file.

Returns a `boolean` value for plug-ins.

SetSource

COMMENT: **OBSOLETE** **SetSource** Netscape

SetSourceSpecifies the URL of the clip to play. This method is backward-compatible with Netscape plug-ins built with RealPlayer version 5.0 or later, but not compatible with version 5.0 ActiveX controls

For More Information: You can also use the `SRC` parameter defined on page 111, to specify the URL of the presentation, in the tag definition.

```
SetSource(string source)
```

source
The URL of the clip to play. The source URL can begin with `rtsp://`, `http://`, `pnm://`, or `file://`.

Returns a `boolean` value for plug-ins.

SetTitle

Sets the current clip's title string, overriding any existing title information. `GetTitle` subsequently returns this new value.

```
SetTitle(string title)
```

title
The title string to be set. This title string overrides all subsequent title information in a multiclip presentation.

Returns a `boolean` value for plug-ins.

Netscape OnlySetVolume

Sets the volume level.

`SetVolume(int16 volume)`

volume

The volume level to be set. Valid values are 0 through 100.

> **Warning!** The use of this method varies slightly between programming languages. In C++, this function requires an `int16` parameter, while in Java, an `int32` parameter is required. Either integer type may be used when developing in JavaScript or VBScript.

Returns `void`.

SetWantErrors

Sets the error sink.

`SetWantErrors(boolean set)`

set

If set to `true`, the errors are trapped and no error dialogs occur in the player. If set to `false`, error dialogs are displayed in the player.

Returns a `boolean` value for plug-ins.

SetWantKeyboardEvents

Sets whether or not keyboard events are to be sent (that is, it sets whether the `OnKeyDown`, `OnKeyPress`, and `OnKeyUp` callbacks are to be sent).

`SetWantKeyboardEvents(boolean set)`

set

If set to `true`, the keyboard events will be sent. If set to `false` (default), the keyboard events will not be sent.

Returns a `boolean` value for plug-ins.

SetWantMouseEvents

Sets whether or not mouse events are to be sent (that is, whether the `OnLButtonDown`, `OnLButtonUp`, `OnMouseMove`, `OnRButtonDown`, and `OnRButtonUp` callbacks are to be sent).

`SetWantMouseEvents(boolean set)`

set

If set to `true`, the mouse events are sent. If set to `false` (default), the mouse events are not sent.

Returns a `boolean` value for plug-ins.

Netscape OnlyNetscape OnlyNetscape OnlyNetscape Only

APPENDIX B

CALLBACK METHODS

This appendix describes the RealPlayer callback methods sent to inform an application or script that a RealPlayer event has occurred. The callback methods are listed here in alphabetical order and each description contains information about how to use the callback in your Netscpae plug-in or ActiveX control, an example of syntax and usage, and backward compatibilty tips for various API versions.

For More Information: For information about categories of callback methods, such as those used to handle user interactions with your presentation, see Chapter 2 "Using Methods and Callbacks".

OnAuthorChange

Sent when the author string changes.

`OnAuthorChange(string author)`

author
The new author string.

Returns `void`.

OnBuffering

COMMENT: Parameter type differs between APIs - See the Warning description below

Sends a percentage of the buffering that has completed.

`OnBuffering(int32 flags, int32 percent_complete)`

flags
The buffering flags. One of the following values:

- 0 — Buffering start up.
- 1 — Buffering resulting from a seek.

- 2 — Buffering resulting from network congestion.
- 3 — Buffering resulting from resuming after pausing a live presentation.

> **Note:** If you are programming in C++, use the values of the flags found in the BUFFERING_REASON enumerator in `rmacore.h` (supplied with the RealSystem SDK).

percent_complete
The amount of buffering that is complete, in percent.

> **Warning!** The use of this method varies slightly between programming languages. In C++, the datatype of this parameter is **`int32`**, while in Java, the datatype is **`int16`**. Either datatype may be used when developing in JavaScript or VBScript.

Returns `void`.

OnClipClosed

COMMENT: Method name differs between APIs - See the Warning description below

Sent to indicate that no clip is currently opened by the control. This method is compatible with RealPlayer version 5.0 and later.

```
OnClipClosed(void)
```

> **Warning!** The use of this method varies slightly between programming languages. In C++, the name of the method is `OnClipClosed`, while in Java and JavaScript, the name is `onClipClosed`.

Returns `void`.

OnClipOpened

COMMENT: Method name differs between APIs - See the Warning description below

Sent when a clip is opened by the control. This method is compatible with RealPlayer version 5.0 and later.

```
OnClipOpened(string short_clip_name, string url)
```

Warning! The use of this method varies slightly between programming languages. In C++, the name of the method is `OnClipOpened`, while in Java and JavaScript, the name is `onClipOpened`.

short_clip_name
The name of the clip that is opened.

url
The URL of the clip that is opened.

Returns `void`.

OnContacting

Sent when RealPlayer contacts a host.

```
OnContacting(string host_name)
```

host_name
The host name string.

Returns `void`.

OnCopyrightChange

Sent when the copyright string changes.

```
OnCopyrightChange(string copyright)
```

copyright
The new copyright string.

Returns `void`.

OnErrorMessage

Sent when an error occurs.

```
OnErrorMessage(
   int16 severity,
   int32 rma_code,
   int32 user_code,
    string user_string,
    string more_info_url,
    string error
)
```

severity
The error level for the last error. See `GetLastErrorSeverity` on page 54 for more information about severity levels.

rma_code
The RMA error code from the last error. RMA error codes are described in the header file `pnresult.h` in the RealSystem G2 SDK available at **http://www.realnetworks.com/devzone/**. In normal operation, all RealSystem components need to be able to handle the following basic codes that may be returned by the RealSystem system:

- PNR_FAIL — Operation failed.
- PNR_OK — Operation succeeded.
- PNR_UNEXPECTED — Call was unexpected or method is not implemented.

user_code
The user error code from the last error. For more information, see `GetLastErrorUserCode` on page 54.

user_string
The error string from the last error dialog. For more information, see `GetLastErrorUserString` on page 55.

more_info_url
The "more info" URL from the last error. This may be nothing (for example, if there is no "more info" URL).

error
A text description of the error.

Returns `void`.

OnGotoURL

COMMENT: Method name differs between APIs - See the Warning description below

Sent when an URL event is encountered for the RealPlayer clip currently playing. This event occurs only if the **AutoGotoURL** setting is `FALSE`. (This setting is modified either by using the `AUTOGOTOURL` parameter in the `<EMBED>` or `<OBJECT>` tag, or by using the `SetAutoGoToURL` method.)

This method is compatible with RealPlayer version 5.0 and later.

`OnGotoURL(string url, string target)`

> **Warning!** The use of this method varies slightly between programming languages. In C++, the name of the method is `OnGotoURL`, while in Java and JavaScript, the name is `onGoToURL`.

url
: Contains the URL that would have been sent to the browser if **AutoGotoURL** were `TRUE`.

target
: The name of the browser or frame the URL should have been opened in if **AutoGotoURL** were `TRUE`.

Returns `void`.

OnKeyDown

COMMENT: Parameter type differs between APIs - See the Warning description below

Sent when the user presses and holds down a keyboard key. This callback method is only sent when the`set` parameter of the `SetWantKeyboardEvents` (see page 79) method is set to `true`.

`OnKeyDown(int32 flags, int32 key)`

> **Warning!** When programming in Java or JavaScript, the **`flags`** parameter is not available. The proper syntax is: `OnKeyDown(int32 key)`.

flags
: The bit flags for the key press. Windows defines the values for this parameter in the Windows Platform SDK from Microsoft, under the `WM_CHAR` message.

key
: The key code for the key that was pressed and held.

Returns `void`.

OnKeyPress

COMMENT: Parameter type differs between APIs - See the Warning description below

Sent when the user presses and releases a keyboard key. This callback method is only sent when the `set` parameter of the `SetWantKeyboardEvents` (see page 79) method is set to `true`.

```
OnKeyPress(int32 flags, int32 key)
```

> **Warning!** When programming in Java or JavaScript, the **`flags`** parameter is not available. The proper syntax is: `OnKeyPress(int32 key)`.

flags
: The bit flags for the key press. Windows defines the values for this parameter in the Windows Platform SDK from Microsoft, under the `WM_CHAR` message.

key
: The key code for the key that was pressed and released.

Returns `void`.

OnKeyUp

COMMENT: Parameter type differs between APIs - See the Warning description below

Sent when user releases keyboard key. This callback method is only sent when the `set` parameter of the `SetWantKeyboardEvents` (see page 79) method is set to `true`.

```
OnKeyUp(int32 flags, int32 key)
```

> **Warning!** When programming in Java or JavaScript, the **`flags`** parameter is not available. The proper syntax is: `OnKeyUp(int32 key)`.

flags
: The bit flags for the key press. Windows defines the values for this parameter in the Windows Platform SDK from Microsoft, under the `WM_CHAR` message.

key
: The key code for the key the user has released.

Returns `void`.

OnLButtonDown

Sent when the user holds down the left mouse button. This callback method is only sent when the `set` parameter of the `SetWantMouseEvents` (see page 79) method is set to `true`.

```
OnLButtonDown(int32 button_flags, int32 x_pos, int32 y_pos)
```

button_flags

The bit flags for mouse and mouse button events.

The following table lists the possible values for the **button_flags** parameter:

Parameter Values for the Possible Mouse Button Events

Bit Flag Value	Mouse Button Event
MK_LBUTTON	The left mouse button is pressed.
MK_RBUTTON	The right mouse button is pressed.
MK_SHIFT	The Shift key on the keyboard is pressed.
MK_CONTROL	The Ctrl key on the keyboard is pressed.
MK_MBUTTON	The middle mouse button is pressed.

x_pos

The *x* position of the mouse when the left button is pressed.

y_pos

The *y* position of the mouse when the left button is pressed.

Returns `void`.

OnLButtonUp

Sent when the user releases the left mouse button. This callback method is only sent when the `set` parameter of the `SetWantMouseEvents` (see page 79) method is set to `true`.

```
OnLButtonUp(int32 button_flags, int32 x_pos, int32 y_pos)
```

button_flags
: The bit flags for mouse and mouse button events. For a list of possible values for this parameter, see Table , “Parameter Values for the Possible Mouse Button Events,” on page 87.

x_pos
: The *x* position of the mouse when the left mouse button is released.

y_pos
: The *y* position of the mouse when the left mouse button is released.

Returns `void`.

OnMouseMove

Sent when the user moves the mouse. This callback method is only sent when the `set` parameter of the `SetWantMouseEvents` (see page 79) method is set to `true`.

> **Note:** This callback is sent when the operating system notifies the plug-in or ActiveX control that the mouse has moved.

```
OnMouseMove(int32 button_flags, int32 x_pos, int32 y_pos)
```

button_flags
: The bit flags for mouse and mouse button events. For a list of possible values for this parameter, see Table , “Parameter Values for the Possible Mouse Button Events,” on page 87.

x_pos
: The *x* position of the mouse.

y_pos
: The *y* position of the mouse.

Returns `void`.

OnMuteChange

Sent when the volume is muted or unmuted.

```
OnMuteChange(boolean mute)
```

mute
: If `true`, the volume is muted. If `false`, the volume is restored.

Returns `void`.

OnPlayStateChange

COMMENT: Parameter type differs between APIs - See the Warning description below

Sent when the play state of the presentation in RealPlayer changes.

`OnPlayStateChange(int32 old_state, int32 new_state)`

> **Warning!** When programming an ActiveX control, the **`old_state`** parameter is not available. The proper syntax is: `OnPlayStateChange(int32 new_state)`. If your ActiveX application requires both the `old_state` and `new_state` parameters, use the **OnStateChange** on page 93 callback instead.

old_state
The previous play state.

new_state
The current play state.

The following table lists the possible values for the **old_state** and **new_state** parameters:

Parameter Values for the Possible Play States

Parameter Value	Play State
0	Stopped
1	Contacting
2	Buffering
3	Playing
4	Seeking
5	Paused

Returns `void`.

OnPosLength

Sent when the position in the clip changes.

`OnPosLength(int32 pos, int32 len)`

> **Note:** This callback is intended for a Netscape plug-in only. If you are coding an ActiveX control, use the **OnPositionChange** on page 90 callback instead.

pos
: The current position of the clip, in milliseconds.

len
: The length of the clip, in milliseconds.

Returns `void`.

OnPositionChange

Sent when the position in the clip changes.

```
OnPositionChange(int32 pos, int32 len)
```

> **Note:** This callback is intended for an ActiveX control only. If you are coding a Netscape plug-in, use the **OnPosLength** on page 89 callback instead.

pos
: The current position of the clip, in milliseconds.

len
: The length of the clip, in milliseconds.

Returns `void`.

OnPostSeek

Sent when a seek completes.

```
OnPostSeek(int32 old_time, int32 new_time)
```

old_time
: The presentation time, in milliseconds, before the seek occurred.

new_time
: The presentation time, in milliseconds, after the seek occurred.

Returns `void`.

OnPreFetchComplete

Sent when the component has fetched the stream header information. Called if `PREFETCH` is set to `true` in the `<EMBED>` or `<OBJECT>` tag or if the set parameter of the SetPreFetch (see page 76) method is set to `true`.

```
OnPreFetchComplete(void)
```

Returns `void`.

OnPreSeek

Sent when the user performs a seek by moving the presentation position slider.

```
OnPreSeek(int32 old_time, int32 new_time)
```

old_time
The presentation time, in milliseconds, when the seek occurred.

new_time
The time, in milliseconds, to which the presentation is seeking.

Returns `void`.

OnPresentationClosed

Sent when the presentation stops.

```
OnPresentationClosed(void)
```

Returns `void`.

OnPresentationOpened

Sent when the presentation starts.

```
OnPresentationOpened(void)
```

Returns `void`.

OnRButtonDown

Sent when the user holds down the right mouse button. This callback method is only sent when the `set` parameter of the `SetWantMouseEvents` (see page 79) method is set to `true`.

`OnLButtonDown(int32 button_flags, int32 x_pos, int32 y_pos)`

button_flags
The bit flags for mouse and mouse button events. For a list of possible values for this parameter, see Table , "Parameter Values for the Possible Mouse Button Events," on page 87.

x_pos
The *x* position of the mouse when the right button is pressed.

y_pos
The *y* position of the mouse when the right button is pressed.

Returns `void`.

OnRButtonUp

Sent when the user releases the right mouse button. This callback method is only sent when the `set` parameter of the `SetWantMouseEvents` (see page 79) method is set to `true`.

`OnRButtonUp(int32 button_flags, int32 x_pos, int32 y_pos)`

button_flags
The bit flags for mouse and mouse button events. For a list of possible values for this parameter, see Table , "Parameter Values for the Possible Mouse Button Events," on page 87.

x_pos
The *x* position of the mouse when the right mouse button is released.

y_pos
The *y* position of the mouse when the right mouse button is released.

Returns `void`.

OnShowStatus

COMMENT: Method name differs between APIs - See the Warning description below

Sent to indicate that the status text is changing. This method is compatible with RealPlayer version 5.0 and later.

`OnShowStatus(string status_text)`

Warning! The use of this method varies slightly between programming languages. In C++, the name of the method is

`OnShowStatus`, while in Java and JavaScript, the name is `onShowStatus`.

status_text
The new status text.

Returns `void`.

OnStateChange

Sent when the play state of the presentation in RealPlayer changes.

Note: This callback is intended for an ActiveX control only. If you are coding a Netscape plug-in, use the **OnPlayStateChange** on page 89 callback instead.

```
OnStateChange(int32 old_state, int32 new_state)
```

old_state
The previous play state.

new_state
The current play state.

For a list of possible values for the **old_state** and **new_state** parameters, see Table , "Parameter Values for the Possible Play States," on page 89.

Returns `void`.

OnTitleChange

Sent when the title string changes.

```
OnTitleChange(string title)
```

title
The new title string.

Returns `void`.

OnVolumeChange

COMMENT: **Parameter type differs between APIs - See the Warning description below**

Sent when the volume level changes.

```
OnVolumeChange(int32 new_volume)
```

new_volume
: The new volume level. The valid volume range is 0 through 100, where 0 represents no volume.

> **Warning!** The use of this method varies slightly between programming languages. In C++, the datatype of this parameter is **`int16`**, while in Java, the datatype is **`int32`**. Either datatype may be used when developing in JavaScript or VBScript.

Returns `void`.

APPENDIX

C

TAG PARAMETERS

This appendix identifies the parameters that can be included in the tag definition of your RealPlayer Netscape plug-in or ActiveX control.

Using Netscape Plug-in <EMBED> Tag Parameters

You can include parameters in a Netscape plug-in `<EMBED>` tag definition using this syntax:

```
<EMBED SRC=”...” WIDTH=... HEIGHT=... PARAMETER=value ... PARAMETER=value>
```

and by following these guidelines:

- Parameter names can be any case,
  although this manual depicts them in uppercase.
- Parameter values are not case-sensitive, however,
  file names are typically lowercase.
- Parameter values do not need to be enclosed in quotation marks, except when the value is a URL.

Using ActiveX Control <OBJECT> Tag Parameters

You can include parameters in an ActiveX control `<OBJECT>` tag definition using this syntax:

```
<OBJECT ID=... CLASSID=“...” WIDTH=... HEIGHT=...>
<PARAM NAME=“name” VALUE=“value”>
<...>
<PARAM NAME=“name” VALUE=“value”>
</OBJECT>
```

and by following these guidelines:

- `The PARAM`, `NAME`, and `VALUE` markers can be any case.

- Parameter names can be any case,
  although this manual depicts them in uppercase.
- Parameter values are not case-sensitive, however,
  file names are typically lowercase.
- Parameter values must always be enclosed in double quotation marks.

AUTOGOTOURL

Specifies how to handle URLs embedded in a presentation.

Value(s)	`true` or `false`
Default Value	`true`
Compatibility	RealPlayer 5 or later

When set to `true`, `AUTOGOTOURL` passes all URLs embedded in your presentation to the browser. When set to `false`, RealPlayer sends the URLs to a Java applet or other application via the `OnGotoURL` callback.

For More Information: Beginning with RealPlayer G2, you can also use the `SetAutoGoToURL` method defined on page 66, to dynamically change how embedded URLs are handled at any time.

If you do not include this parameter in your tag definition and your presentation contains embedded URLs, then the URLs are not passed to the browser (default).

AUTOSTART

Specifies whether to automatically play a presentation.

Value(s)	`true` or `false`
Default Value	`true`
Compatibility	RealPlayer 5 or later

When set to `true`, `AUTOSTART` instructs the Embedded RealPlayer to automatically begin playing the presentation once the source content is available. If multiple controls have been linked together using the `CONSOLE` parameter, `AUTOSTART` can be set to `true` in one or more tag definitions. When set to `false`, the presentation is not automatically played back.

> **For More Information:** If you are developing a Netscape plug-in in RealPlayer version 5.0 or later, you can also use the `SetAutoStart` method defined on page 66, to dynamically control automatic playback at any time.

If you do not include this parameter in your tag definition, then your presentation will not automatically start (default).

BACKGROUNDCOLOR

Sets the background color for the image window. When a clip includes transparent regions, the background color also shows through these areas.

Value(s)	color name or RGB hex value
Default Value	`black`
Compatibility	RealPlayer G2 or later

The background color is specified using an RGB hexadecimal color value (`#RRGGBB`) or a color name. The following table lists the valid background color values:

<TD bgcolor=#FFFFFF> <FONT COLOR="#000000">white<BR> (#FFFFFF)</FONT> </TD>	<TD bgcolor=#C0C0C0><FONT COLOR="#000000">silver<BR> (#C0C0C0)</FONT></TD>	<TD bgcolor=#808080><FONT COLOR="#FFFFFF">gray<BR> (#808080)</FONT></TD>	<TD bgcolor=#000000> <FONT COLOR="#FFFFFF">black<BR> (#000000)</FONT></TD>

<TD bgcolor=#FFFF00> <FONT COLOR="#000000">yellow<BR> (#FFFF00)</FONT></TD>	<TD bgcolor=#FF00FF> <FONT COLOR="#FFFFFF">fuchsia<BR> (#FF00FF)</FONT></TD>	<TD bgcolor=#FF0000><FONT COLOR="#FFFFFF">red<BR> (#FF0000)</FONT></TD>	<TD bgcolor=#800000> <FONT COLOR="#FFFFFF">maroon<BR> (#800000)</FONT></TD>
<TD bgcolor=#00FF00> <FONT COLOR="000000">lime<BR> (#00FF00)</FONT></TD>	<TD bgcolor=#808000> <FONT COLOR="#FFFFFF">olive<BR> (#808000)</FONT></TD>	<TD bgcolor=#008000><FONT COLOR="#FFFFFF">green<BR> (#008000)</FONT></TD>	<TD bgcolor=#800080> <FONT COLOR="#FFFFFF">purple<BR> (#800080)</FONT></TD>
<TD bgcolor=#00FFFF> <FONT COLOR="#000000">aqua<BR> (#00FFFF)</FONT></TD>	<TD bgcolor=#008080> <FONT COLOR="#FFFFFF">teal<BR> (#008080)</FONT></TD>	<TD bgcolor=#0000FF><FONT COLOR="#FFFFFF">blue<BR> (#0000FF)</FONT></TD>	<TD bgcolor=#000080> <FONT COLOR="#FFFFFF">navy<BR> (#000080)</FONT></TD>

For More Information: You can also use the `SetBackgroundColor` method defined on page 67, to dynamically change the background color of the image window at any time.

If you do not include the `BACKGROUNDCOLOR` parameter in your tag definition, the background color for the image window is set to black (default).

CENTER

Specifies whether the presentation should be centered in the image window and displayed in its original, encoded size.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer G2 or later

When the `CENTER` parameter is set to `true`, the presentation is centered in the image window and the height and width of the presentation are reset to the original dimensions (specified by the `WIDTH` and `HEIGHT` parameters when the embedded presentation was encoded). When `CENTER` is set to `false`, the presentation is not centered and the height and width of the presentation are allowed to expand and fill the image window.

For More Information: You can also use the `SetCenter` method defined on page 68, to dynamically center the presentation in the image window at any time.

If you do not include the `CENTER` parameter in your tag definition, then the presentation is not centered or resized to its original dimensions (default).

Warning! The `CENTER` and `MAINTAINASPECT` parameters cannot both be set to `true`. In addition, the set methods for these parameters (`SetCenter` and `SetMaintainAspect`) also cannot both be `true`. When one parameter or set method is set to `true`, the other parameter and set method are considered to be `false`.

CLASSID

Identifies an ActiveX control as belonging to the RealPlayer class.

Value(s)	`clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA`
Default Value	(none)
Compatibility	ActiveX only, RealPlayer 5 or later

An Embedded RealPlayer ActiveX control must include the RealPlayer classID value in the `<OBJECT>` tag definition and the value must be enclosed in double quotation marks:

```
<OBJECT ID=... CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA" ... >
```

If you do not include this parameter in your tag definition, or you specify an invalid value, the browser will not load your presentation and may issue an error message.

For More Information: For more information about creating an embedded presentation using an ActiveX control, see "Using <OBJECT> Tags for the ActiveX Control" on page 9.

CONSOLE

Specifies whether multiple controls should be linked together to manage playback of a single embedded presentation.

Value(s)	shared name, `_master`, or `_unique`
Default Value	`_unique`
Compatibility	RealPlayer 5 or later

When the same console name is specified for multiple control on a Web page, this parameter can be used to enable these controls to manage playback of a single embedded presentation. For example, if you have multiple **Play** and **Stop** buttons on the same page, a shared console name enables them to control the same clip. The following table lists the valid console name values:

shared name	Links the control with other Embedded RealPlayer controls which have the same **shared name** value in their tag definitions. For example, "`console1`".
`_master`	Links the control with all other Embedded RealPlayer controls.
`_unique`	Does not link the control with any other Embedded RealPlayer controls.

For More Information: You can also use the `SetConsole` method defined on page 69, to dynamically specify whether your controls are linked at any time. For more information about linking multiple controls, see the *RealSystem Production Guide*.

If you do not include this parameter in your tag definition, then the console name of the control is considered to be `_unique` (default).

CONTROLS

Embeds the specified RealPlayer control on your Web page.

Value(s)	control name
Default Value	`All`
Compatibility	RealPlayer 5 or later

For a complete listing of valid control name values, see “Appendix D: RealPlayer Controls” beginning on page 113.

> **For More Information:** You can also use the `SetControls` method defined on page 70, to dynamically add controls to your Web page at any time.

If you do not include this parameter in your tag definition, then the basic RealPlayer control panel is added to your Web page (default - `All`).

HEIGHT

Sets the height of the image window or a specified embedded control.

Value(s)	number of pixels or percentage of browser window size
Default Value	(none)
Compatibility	RealPlayer 5 or later

Setting the `WIDTH` and `HEIGHT` parameters to zero causes the control to be hidden. If you do not include this parameter in your image window tag definition, the window may appear as a tiny icon because streaming media presentations do not size automatically.

ID

Identifies the embedded presentation for referencing in VBScript.

Value(s)	any unique identification string
Default Value	(none)
Compatibility	ActiveX only, RealPlayer 5 or later

When you intend to use scripting to control your presentation, you must specify a unique value for the `ID` parameter, such as `ID=RVOCX`. After your presentation has been identified in an `<OBJECT>` tag similar to the following:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA">
<PARAM NAME="SRC" VALUE="file://my_content.rpm">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
...
</OBJECT>
```

you can then use VBScript, or any programming language supported by the browser, to issue RealPlayer commands to control the presentation:

```
<FORM>
<input TYPE="button" VALUE="Play" NAME="doplay">
  <script LANGUAGE="VBScript" FOR="doplay" EVENT="onClick">
  RVOCX.DoPlay
  </script>
</FORM>
```

For More Information: For more information about creating an embedded presentation using an ActiveX control, see “Using <OBJECT> Tags for the ActiveX Control” on page 9.

IMAGESTATUS

Specifies whether the status information should be written along the bottom of the image window.

Value(s)	`true` or `false`
Default Value	`true`
Compatibility	RealPlayer 5 or later

When set to `true`, or if the parameter is not included in the tag definition, then the status information is written along the bottom of the image window (default). When set to `false`, the status information is not displayed.

For More Information: You can also use the `SetImageStatus` method defined on page 73, to dynamically change whether the status information is displayed at any time.

LOOP

Specifies whether playback of the clip should continue, or *loop*, indefinitely.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer G2 or later

When set to `true`, playback of the clip continues to loop until the user stops the presentation. If multiple controls have been linked together using the `CONSOLE` parameter, `LOOP` can be set to `true` in one or more tag definitions.

For More Information: You can also use the `SetLoop` method defined on page 73, to dynamically change whether the clip should loop at any time.

When `LOOP` is set to false, or if the parameter is not included in the tag definition, then the presentation stops after the first playback (default).

MAINTAINASPECT

Specifies whether the height-to-width (aspect) ratio of the clip should stay constant when the clip scales to fit the image window.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer G2 or later

If the `MAINTAINASPECT` parameter is set to `true`, the aspect ratio of the clip remains constant when the image window is resized. When this occurs, the clip is centered in the image window and scaled until one dimension reaches the window's boundaries and the other dimension is within the boundaries. If multiple controls have been linked together using the `CONSOLE` parameter, `MAINTAINASPECT` can be set to `true` in one or more tag definitions.

For More Information: You can also use the `SetMaintainAspect` method defined on page 73, to dynamically change whether the correct aspect ratio should be maintained at any time.

If the `MAINTAINASPECT` parameter is set to `false`, or if it is not included in the tag definition, then the dimensions of the clip scale as necessary to allow the clip to completely fill the image window (default). When the dimensions of the clip are allowed to change in this manner, the source image may appear distorted.

Warning! The `MAINTAINASPECT` and `CENTER` parameters cannot both be set to `true`. In addition, the set methods for these parameters (`SetMaintainAspect` and `SetCenter`) also cannot both be `true`. When one parameter or set method is set to `true`, the other parameter and set method are considered to be `false`.

NAME

Specifies the name to associate with an embedded RealPlayer control, to enable the control to be referenced using JavaScript.

Value(s)	any unique name
Default Value	(none)
Compatibility	Netscape only, RealPlayer 5 or later

In order to refer to an embedded RealPlayer control from JavaScript, you must specify a name for the control in the `NAME` parameter in the `<EMBED>` tag definition. For example:

```
<EMBED NAME=my_name SRC="..." WIDTH=176 HEIGHT=132>
```

The control can then be referenced by a JavaScript command, such as the following:

```
<Input Type="button" Value="play" onClick="document.my_name.DoPlay()">
```

If you are using more than one instance of a single type of embedded control or a variety of different embedded controls in your Web page, then each instance must have a unique `NAME`. Using different names for each instance ensures that you can use JavaScript to manage each embedded control individually, if necessary.

Warning! In Netscape versions 4.x, you can only reference named controls from JavaScript, when the `NOJAVA` parameter is *not* set to `true`. If `NOJAVA=true` is included in your `<EMBED>` tag, the browser's Java Virtual Machine (JVM) is prevented from starting if it is not yet running, and control referencing from JavaScript is not available.

NOJAVA

Prevents the Java Virtual Machine (JVM) from starting, if it is not yet running, and makes the use of JavaScript impossible.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	Netscape version 4.x only, RealPlayer G2 or later

Setting `NOJAVA=true` prevents the JVM from starting, if it is not yet running, thus prohibiting `NAMED` controls from being referenced using JavaScript.

When `NOJAVA` is set to `false`, or if the parameter is not included in your control tag definition, the JVM is started and `NAMED` controls can be referenced from JavaScript (default). However, because the other parameters described in this chapter do not require the JVM and starting the JVM delays presentation playback, it is highly recommended that you specify `NOJAVA=true` in the tag definition for every control, when you do not intend to use scripting.

Note: Although you can specify `NOJAVA` in an ActiveX `<OBJECT>` tag or in a Netscape 6.0 plug-in, doing so has no effect because Internet Explorer and Netscape Navigator 6.0 launch the JVM on browser start-up.

NOLABELS

Specifies whether the title, author, and copyright information for a presentation should be displayed in RealPlayer. This parameter is obsolete and should only be used to provide backward compatibility with RealPlayer 5.

Value(s)	`true` or `false`

Default Value	`false`
Compatibility	Netscape only, RealPlayer 5 or later

When the `NOLABELS` parameter is set to `true`, the title, author, and copyright information is not displayed. When `NOLABELS` is set to `false`, or if you do not include the parameter in your tag definition, the presentation information is displayed in RealPlayer (default).

For More Information: You can also use the `SetNoLabels` method defined on page 74, to dynamically specify whether the presentation information should be displayed at any time.

NOLOGO

Specifies whether the RealNetworks logo should be displayed in the image window when no clip is playing.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer G2 or later

When the `NOLOGO` parameter is set to `true`, the RealNetworks logo is not displayed in the image window when no clip is playing. The image window then defaults to black unless a `BACKGROUNDCOLOR` has been specified.

For More Information: You can also use the `SetNoLogo` method defined on page 75, to dynamically specify whether the logo should be displayed at any time.

When `NOLOGO` is set to `false`, or if you do not include the parameter in your tag definition, the RealNetworks logo is displayed in the image window when no clip is playing (default).

NUMLOOP

Specifies the number of the times the presentation should loop during playback.

Value(s)	any number
Default Value	(none)
Compatibility	RealPlayer G2 or later

When the `NUMLOOP` parameter has been set to a number value, such as `2`, the presentation loops (plays from beginning to end) the specified number of times and then stops.

> **For More Information:** You can also use the `SetNumLoop` method defined on page 75, to dynamically specify the number of times the presentation should loop at any time.

If you do not include the `NUMLOOP` parameter in your tag definition (default), then the presentation only loops if the `LOOP` parameter has been specified.

> **Warning!** If both the `NUMLOOP` and `LOOP` parameters have been specified, or both of the set methods for these parameters (`SetLoop` and `SetNumLoop`) have been used, the `LOOP` parameter / method is ignored. This condition still applies even if `NUMLOOP` has been set to zero.

PARAM

Used to specify additional parameters in an ActiveX control `<OBJECT>` tag definition.

Value(s)	any parameter described in this chapter, except `NAME`
Default Value	(none)
Compatibility	ActiveX only, RealPlayer 5 or later

Additional parameters are specified via the `PARAM` parameter, using this syntax:

```
<PARAM NAME="name" VALUE="value">
```

The `NAME` variable can be assigned any of the parameters described in this chapter, except for the NAME parameter. (To specify a name for a control in your ActiveX control `<OBJECT>` tag defintion, use the ID parameter instead, defined on page 9.) The `VALUE` variable should be assigned the appropriate value for the parameter specified in `NAME`.

> **For More Information:** For more information about including additional parameters in your ActiveX control, see "Using ActiveX Control <OBJECT> Tag Parameters" on page 95.

PREFETCH

Enables or disables `PREFETCH` playback mode, which causes RealPlayer to get the stream description information from a presentation before playback begins.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer 5 or later

When RealPlayer detects that prefetch playback mode is enabled, the presentation's stream description information is obtained. After the information has been captured, `OnPreFetchComplete` (see page 91) is returned to the plug-in or control and the presentation is paused. By setting the `PREFETCH` parameter to `true`, you can obtain information about a presentation before it begins playing, and use the information to change control or playback characteristics.

For example, after the description information has been fetched, you could find out the size and width of an embedded clip using `GetClipWidth` and `GetClipHeight` (see page 47). You could then dynamically create the `<EMBED>` or `<OBJECT>` tag for the image window using the clip's native size for the `WIDTH` and `HEIGHT` parameters.

> **For More Information:** You can also use the `SetPreFetch` method defined on page 76, to dynamically specify whether prefetch playback mode is enabled at any time.

When PREFETCH is set to false, or if you do not include the parameter in your tag definition, then prefetch playback mode is disabled (default).

REGION

Defines the Web page playback region for a SMIL clip.

Value(s)	SMIL region
Default Value	(none)
Compatibility	RealPlayer G2 or later

Clips listed in a SMIL file include specifications in their tags, which indicate where the clip should be played back in a Web page. For example, if the newsarticle.rt file should be displayed in the article playback region of a newspaper Web page, it is listed in the newspaper.smil file as:

```
<textstream src="newsarticle.rt" region="article " />
```

In order to view newsarticle.rt on the newspaper Web page, an image window control has to be created and it's playback region has to be set to article using the REGION parameter using a Netscape plug-in:

```
<EMBED SRC="http://realserver.example.com:8080/ramgen/newspaper.smil?embed"
       WIDTH=176 HEIGHT=132 NOJAVA=true CONTROLS=ImageWindow
        REGION=article>
```

or an ActiveX control:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
        WIDTH=176 HEIGHT=132>
<PARAM NAME="SRC"
       VALUE="http://realserver.example.com:8080/ramgen/newspaper.smil">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="REGION" VALUE="article">
</OBJECT>
```

You can define similar <EMBED> or <OBJECT> tags to create other regions for other clips in the SMIL file, and each tag lists the same SMIL file in the SRC parameter.

> **Warning!** When the layout of a presentation is defined using HTML, the <layout> section of the SMIL file header must be omitted.

COMMENT: For more information about creating SMIL presentations, see ???.

SCRIPTCALLBACKS

Specifies the callback events to handle via a comma separated list.

Value(s)	any callback described in Appendix B beginning on page 81 or `All`
Default Value	(none)
Compatibility	Netscape 6.0 only, RealPlayer 5 or later

The `SCRIPTCALLBACKS` parameter can be used by Netscape version 6.0 plug-ins, to specify the set of callback events you would like to capture and handle. The events are assigned to the parameter via a comma separated list, such as:

```
<SCRIPTCALLBACKS=OnPresentationOpened,OnPresentationClosed>
```

Any of the callbacks described in Appendix B beginning on page 81 can be included in the list, or you can specify `All` to capture all events.

For More Information: For more information about setting up event handling in your Netscape plug-in, see "Handling Events in Netscape Version 6.0" on page 12.

SHUFFLE

Specifies whether all unplayed clips in a presentation, should be played back in a random order. This parameter can be used for multiclip RAM files (`.ram` or `.rpm`) or SMIL files that contain only a sequence of clips.

Value(s)	`true` or `false`
Default Value	`false`
Compatibility	RealPlayer G2 or later

When the `SHUFFLE` parameter is set to `true`, all unplayed clips in a presentation are played back in a random order, rather than in the order in which they appear in the file.

For More Information: You can also use the `SetShuffle` method defined on page 77, to dynamically specify whether clip playback should be randomized at any time.

When `SHUFFLE` is set to `false`, or if you do not include the parameter in your tag definition, the clips are played back in the order in which they appear in the multiclip RAM file or SMIL file (default).

SRC

Specifies the URL of the presentation to be played.

Value(s)	URL
Default Value	(none)
Compatibility	RealPlayer 5 or later

The URL can begin with `rtsp://`, `http://`, `pnm://`, or `file://` and the entire URL string must be enclosed in double quotation marks. The URL source content is specified such as:

```
<EMBED SRC="http://.../.../my_content.rpm" WIDTH=176 HEIGHT=132>
```

Any directories or file specified in the URL cannot contain spaces in their names.

For More Information: You can also use the `SetSource` method defined on page 78, to dynamically specify the URL of the presentation at any time.

The `SRC` parameter may be omitted from the tag definition, when the content mime `TYPE` parameter is specified similar to:

```
<EMBED WIDTH=176 HEIGHT=132 TYPE="audio/x-pn-realaudio-plugin">
```

however, doing so may produce unexpected results. Therefore, it is strongly recommended that you always include the `SRC` parameter and at least supply the name of an empty presentation file.

TYPE

Identifies the MIME type of the presentation specifed in the `SRC` parameter, defined on page 111.

Value(s)	MIME type
Default Value	(none)
Compatibility	RealPlayer 5 or later

The typical syntax for the MIME type looks like the following:

```
<EMBED SRC="http://.../.../my_content.rpm" WIDTH=176 HEIGHT=132
        TYPE="audio/x-pn-realaudio-plugin">
```

The browser reads the MIME type value and then embeds the appropriate plug-in for the presentation. If you do not include this parameter in your tag definition, the browser may not load the ideal plug-in for your presentation.

> **Tip:** When the `TYPE` parameter is specified, the `SRC` parameter may be omitted from the tag definition. However, doing so may produce unexpected results. Therefore, it is strongly recommended that you always include the `SRC` parameter and at least supply the name of an empty presentation file.

WIDTH

Sets the width of the image window or a specified embedded control.

Value(s)	number of pixels or percentage of browser window size
Default Value	(none)
Compatibility	RealPlayer 5 or later

Setting the `HEIGHT` and `WIDTH` parameters to zero causes the control to be hidden. If you do not include this parameter in your image window tag definition, the window may appear as a tiny icon because streaming media presentations do not size automatically.

APPENDIX

D

REALPLAYER CONTROLS

This appendix identifies the RealPlayer controls that can be added to your Web page using the `CONTROLS` parameter, defined on page 100. Each description includes the parameter value that must be specified to create the control, the recommended width and height to reproduce the control as it appears in RealPlayer, and an example of how the control will look on your Web page.

COMMENT:ToDo - Insert RealPlayer 9 screenshots after BETA release

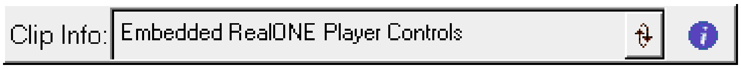
RealPlayer Controls

RealPlayer controls are individual components that support specific RealPlayer functionality, such as playing a presentation, muting clip volume, and displaying the current clip position. RealPlayer controls are also sometimes available in combination with other controls, to support different scenarios of playback control and presentation status.

For More Information: For more information about embedding combinations of playback and status controls, see "Convenience Controls" on page 117.

Clip Information Field

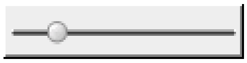
An information area that displays the title, author, and copyright for the currently playing clip or portion of a multi-clip. The arrow button to the right of the field allows you to scroll through the information. The round button opens a dialog with additional information about the clip, supplied by the content provider.



`CONTROLS` **Value**	`TACCtrl`
Width	370 pixels
Height	32 pixels

Clip Position Slider

A slide control that indicates the currently playing position within the clip. You can jump to other positions within the clip by dragging the slider to the left or the right.

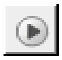


`CONTROLS` **Value**	`PositionSlider`
Width	120 pixels
Height	26 pixels

Note: When the clip is a live broadcast, the clip position slider is automatically disabled.

Fast Forward Button

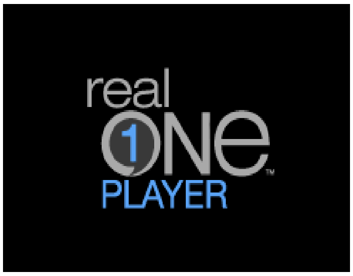
A button with a graphical image representing the fast forward action. When this button is clicked, playback of the currently loaded clip jumps to the next frame. If the button is pressed down, playback continues to jump forward until the button is released.



`CONTROLS` **Value**	`FFCtrl`
Width	28 pixels
Height	18 pixels

Image Window

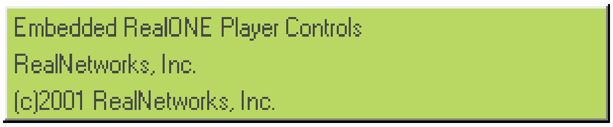
A basic image window for displaying video or animation presentations. While the presentation is loading, the RealPlayer G2 logo is displayed in the image window. Even when no other controls are visible on the Web page, the user can typically right-click (in Windows) or hold down the mouse button (on Macintosh) over the window to display a menu of playback choices, such as **Play** and **Stop**.



`CONTROLS` **Value**	`ImageWindow`
Width	176 pixels or wider, depending on your presentation width
Height	132 pixels or higher, depending on your presentation height

Information Panel

An information area that displays the title, author, and copyright for the currently playing presentation.

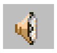


Embedded RealONE Player Controls
RealNetworks, Inc.
(c)2001 RealNetworks, Inc.

`CONTROLS` **Value**	`InfoPanel`
Width	300 pixels
Height	55 pixels

Mute Button

A button with a graphical image representing the mute volume action. When the mute action has been selected, the button appears to be pressed in and the clip volume is temporarily lowered to zero. Clicking the button again, deselects the mute action and restores the clip volume to its previous level.

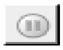


`CONTROLS` **Value**	`MuteCtrl`

Width	26 pixels
Height	26 pixels

Pause Button

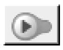
A button with a graphical image representing the pause action. When this button is clicked, playback of the currently loaded clip is paused. If the clip is already paused or stopped, clicking this button has no effect.



`CONTROLS` **Value**	`PauseButton`
Width	26 pixels
Height	26 pixels

Play Button

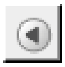
A button with a graphical image representing the play (start) action. When this button is clicked, playback of the currently loaded clip is started. If the clip is already playing, clicking this button has no effect.



`CONTROLS` **Value**	`PlayOnlyButton`
Width	26 pixels
Height	26 pixels

Rewind Button

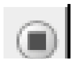
A button with a graphical image representing the rewind action. When this button is clicked, playback of the currently loaded clip jumps to the previous frame. If the button is pressed down, playback continues to jump backward until the button is released.



`CONTROLS` **Value**	`RWCtrl`
Width	28 pixels
Height	18 pixels

Stop Button

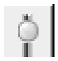
A button with a graphical image representing the stop action. When this button is clicked, playback of the currently loaded clip is halted and reset to the beginning. If the clip is already stopped, clicking this button has no effect.



`CONTROLS` **Value**	`StopButton`
Width	26 pixels
Height	26 pixels

Volume Slider

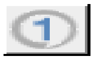
A slider with a graphical image representing the volume control. Dragging the volume button up the slider increases the clip volume, and dragging the button down decreases the volume.



`CONTROLS` **Value**	`MuteVolume`
Width	26 pixels
Height	65 pixels

www.real.com Home Button

A button with the RealNetworks logo. When the button is clicked, a browser window is opened and connected to the **www.real.com** Web site.



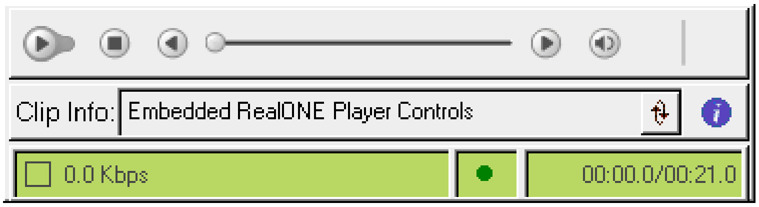
`CONTROLS` **Value**	`HomeCtrl`
Width	45 pixels
Height	45 pixels

Convenience Controls

Convenience controls are combinations of RealPlayer controls that provide different levels of support for controlling playback of your presentation and providing status to your audience.

All Playback Controls

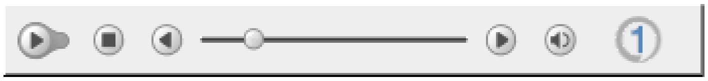
A complex panel of playback and status controls, including the www.real.com Home Button defined on page 117, Play Button defined on page 116, Pause Button defined on page 116, Stop Button defined on page 117, Fast Forward Button defined on page 114, Rewind Button defined on page 116, Clip Position Slider defined on page 114, Clip Information Field defined on page 113, and Status Bar.



`CONTROLS` **Value**	`All` or `default`
Width	375 pixels
Height	100 pixels

Control Panel, Simple

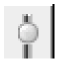
A simple panel of playback controls, including the www.real.com Home Button defined on page 117, Play Button defined on page 116, Pause Button defined on page 116, Stop Button defined on page 117, Fast Forward Button defined on page 114, Rewind Button defined on page 116, and Clip Position Slider defined on page 114.



`CONTROLS` **Value**	`All`
Width	350 pixels
Height	36 pixels

Mute / Volume Bar

A simple playback control bar consisting of the Mute Button defined on page 115, and the Volume Slider defined on page 117.

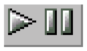


`CONTROLS` **Value**	`MuteVolume`
Width	26 pixels
Height	88 pixels

Play / Pause Bar

A simple playback control bar consisting of the Play Button defined on page 116, and the Pause Button defined on page 116.

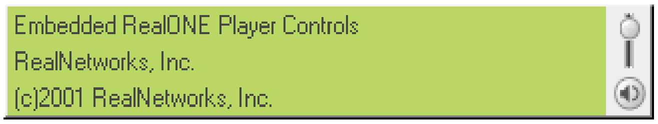
> **Note:** This control is available only for RealPlayer 8 and earlier versions.



`CONTROLS` **Value**	`PlayButton`
Width	44 pixels
Height	26 pixels

Information / Volume Bar

A simple playback control bar consisting of an Information Panel defined on page 115 and the Mute / Volume Bar defined on page 119.



Embedded RealONE Player Controls
RealNetworks, Inc.
(c)2001 RealNetworks, Inc.

`CONTROLS` **Value**	`InfoVolumePanel`
Width	325 pixels
Height	55 pixels

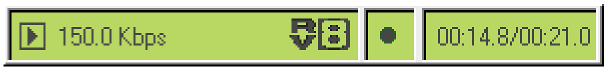
Status Panels

Status panels display information about your presentation using a variety of different controls, including a text message area, a network congestion LED, and a current clip position indicator.

Note: If you do not embed a status panel in your Web page, error messages received from RealPlayer are displayed in the browser's status bar.

Status Bar

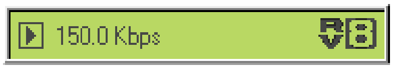
A complex status panel consisting of a text message area, the network congestion LED, and the current clip position indicator.



`CONTROLS` **Value**	`StatusBar`
Width	325 pixels
Height	30 pixels

Status Field

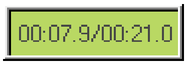
An simple status panel consisting only of a text message area.



`CONTROLS` **Value**	`StatusField`
Width	200 pixels
Height	30 pixels

Position Field

An information panel consisting only of the current clip position indicator, that identifies the clip's current place within the presentation timeline and the total clip length.



`CONTROLS` **Value**	`PositionField`

Width	90 pixels
Height	30 pixels

APPENDIX

E

API VERSION COMPATIBILITY

This appendix identifies the following compatibility information for the Embedded control methods and callback methods:

- Fully compatible with RealPlayer vesion 5.0 and later
- Obsolete in releases later than RealPlayer 8.*x*
- Removed from the Netscape and/or ActiveX API sets

Fully Compatible Methods and Callback Methods

The following methods can be used with RealPlayer vesion 5.0 and later. To support all these versions of RealPlayer, your application can issue only these methods, and will receive only these callback methods:

Methods Compatible With RealPlayer 5.0 and Later

`AboutBox` on page 41	`DoStop` on page 44	`SetAutoGoToURL` on page 66
`CanPlayPause` on page 41	`EditPreferences` on page 44	`OnClipOpened` on page 82
`CanStop` on page 42	`HasNextItem` on page 64	`OnClipClosed` on page 82
`DoNextItem` on page 43	`HasPrevItem` on page 64	`OnGotoURL` on page 84
`DoPlayPause` on page 43	`HideShowStatistics` on page 65	`OnShowStatus` on page 92
`DoPrevItem` on page 44	`Netscape OnlyIsStatisticsVisible` on page 65	

Methods Compatible With RealPlayer 5.0 and Later

DoGotoURL on page 42 (ActiveX Control only)	SetConsoleName on page 69 (Netscape Plug-in only)	Netscape OnlyNetscape OnlySetControlString on page 70 (Netscape Plug-in only)
SetNoLabels on page 74 (Netscape Plug-in only)	SetSource on page 78 (Netscape Plug-in only)	SetAutoStart on page 66 (Netscape Plug-in only)

Methods Removed From the ActiveX and Netscape API Sets

The following methods have been removed from the ActiveX and Netscape APIs and are no longer supported. Calls to any of the following functions will be ignored by the Embedded RealPlayer. No new methods have been introduced to the API sets to replace these removed methods..

Removed Methods

EnableMesssageBox	GetLastLeftButton DownYPos	GetLastRightButton UpTimeStamp
EnableSetPreference	GetLastLeftButton UpKeyFlags	GetLastRightButton UpXPos
GetContextMenu	GetLastLeftButton UpTimeStamp	GetLastRightButton UpYPos
GetContextMenuItem	GetLastLeftButton UpXPos	GetRegion
GetLastErrorRMA CodeString	GetLastLeftButton UpYPos	GetVideoState
GetLastKeyDownKey	GetLastMouseMove KeyFlags	IsDone
GetLastKeyDown TimeStamp	GetLastMouseMove TimeStamp	IsZoomed
GetLastKeyPressKey	GetLastMouseMove XPos	OnStatsInfoChange

Removed Methods

GetLastKeyPress TimeStamp	GetLastMouseMoveYPos	OnStatisticsChanged
GetLastKeyUpKey	GetLastRightButton DblKeyFlags	ProcessIdle
GetLastKeyUp TimeStamp	GetLastRightButton DblTimeStamp	SetContextMenu
GetLastLeftButton DblKeyFlags	GetLastRightButton DblXPos	SetContextMenuItem
GetLastLeftButton DblTimeStamp	GetLastRightButton DblYPos	SetRegion
GetLastLeftButton DblXPos	GetLastRightButton DownKeyFlags	SetVideoState
GetLastLeftButton DblYPos	GetLastRightButton DownTimeStamp	SetZoomed
GetLastLeftButton DownKeyFlags	GetLastRightButton DownXPos	StatusScanEnd
GetLastLeftButton DownTimeStamp	GetLastRightButton DownYPos	StatusScanStart
GetLastLeftButton DownXPos	GetLastRightButton UpKeyFlags	StatusScan

Obsolete Methods in the ActiveX and Netscape API Sets

The following methods are considered to be obsolete, but may be used depending on your development platform and application requirements. In some instances, new methods have been added to the API sets to replace some of the obsolete methods. When this is the case, it is strongly recommended that you use the new method.

Obsolete Methods and Their Replacement Methods

Obsolete Method	Replaced by New Method
AboutBox	ActiveX OnlySetShowAbout

Obsolete Methods and Their Replacement Methods

Obsolete Method	Replaced by New Method
`CanPlayPause`	`CanPlay` and `CanPause`
`DoNextItem`	`DoNextEntry`
`DoPlayPause`	`DoPlay` and `DoPause`
`DoPrevItem`	`DoPrevEntry`
`EditPreferences`	`SetShowPreferences`
`GetNoLabels`	*no replacement*
`HasNextItem`	`HasNextEntry`
`HasPrevItem`	`HasPrevEntry`
`HideShowStatistics`	`SetShowStatistics`
`IsStatisticsVisible`	`GetShowStatistics`
`SetConsoleName`	`SetConsole`
`SetControlString`	`SetControls`
`SetNoLabels`	*no replacement*

INDEX