REALONE PLAYER SCRIPTING GUIDE

Last Update: 15 October 2002

RealNetworks, Inc.
PO Box 91123
Seattle, WA 98111-9223
U.S.A.

http://www.real.com
http://www.realnetworks.com

©2002 RealNetworks, Inc. All rights reserved.

Information in this document is subject to change without notice. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of RealNetworks, Inc.

Printed in the United States of America.

Helix, The Helix Logo, RBN, the Real "bubble" (logo), Real Broadcast Network, RealAudio, Real.com, RealJukebox, RealMedia, RealNetworks, RealPlayer, RealOne, RealPresenter, RealSlideshow, RealSystem, RealText, RealVideo, SureStream, and Surreal.FX Design are trademarks or registered trademarks of RealNetworks, Inc.

Other product and corporate names may be trademarks or registered trademarks of their respective companies.

CONTENTS

INTRODUCTION

This guide will help you to use Javascript or ActiveX in two areas. First, it explains methods that you can use to coordinate streaming media and HTML pages in the various RealOne Player panes. Second, it shows how to use scripting to extend the functionality of the embedded player, a variation of RealOne Player that enables streaming media to play directly within a Web page.

> **Note:** Although this guide shows you how to use Javascript and ActiveX with RealOne Player, it assumes that you are already familiar with these scripting technologies in general.

What is Helix?

Helix™ from RealNetworks is a universal digital media delivery platform. With industry-leading performance, integrated content distribution, advertising, user authentication, Web services support, and native delivery of RealMedia, Windows Media, QuickTime, and MPEG-4, Helix from RealNetworks is a robust digital media foundation that meets the needs of enterprises and networking service providers.

System Components

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

- RealOne Player

  Use RealOne Player, available free at **http://www.real.com**, to test your Javascript or ActiveX extensions. The RealOne Player installation includes the Netscape plug-in and ActiveX control for embedded presentations.

- Helix Server

Helix Server streams clips to RealPlayer. The server is not necessary for testing local playback of clips, but it is necessary for streaming presentations over a network. The *Helix Server Administration Guide* is available at the following Web address:

http://service.real.com/help/library/servers.html

- Software Development Kit (SDK)

  The SDK is not necessary for using RealOne Player's scriptable playback features. But it is required for advanced programming tasks, such as building a new client interface on top of the RealOne Player core. A knowledge of C++ programming is required to use the SDK. Register for and download the SDK from this Web page:

  http://proforma.real.com/rnforms/resources/server/realsystemsdk/index.html

How to Download This Guide to Your Computer

RealNetworks makes this guide available in the following formats for download to your computer:

- The HTML+Javascript version is available as a single, zipped archive that includes samples that you can play in RealOne Player. You can read this version with Netscape Navigator or Microsoft Internet Explorer.
- The HTML Help version is available as a single `.chm` file for Windows 98 and later operating systems. It is identical to the HTML+Javascript version, except that it does not contain any sample files. The HTML Help version is smaller in size than the HTML+Javascript version, and it includes a search function.
- An Adobe Acrobat (PDF) version includes page numbers in cross-references, making it more useful than the HTML versions when printed. You can download the free Acrobat viewer from Adobe's Web site at **http://www.adobe.com/products/acrobat/readstep.html**.

All of the online versions of this guide are available for individual download from RealNetworks' Technical Support Web site at:

http://service.real.com/help/library/encoders.html

How This Guide Is Organized

Part I: Scripting with RealOne Player

The chapters in this section explain how to use Javascript and ActiveX in the RealOne Player environment, in which RealOne Player launches as a separate application.

Chapter 1: RealOne Player Environment

This chapter explains the RealOne Player environment, covering the three-pane design. It introduces you to the variety of authoring methods you can use, and explains the functional areas in which you can use Javascript or ActiveX.

Chapter 2: RealOne Player Methods

This chapter lists the Javascript and ActiveX methods that you can use to control playback in the RealOne Player environment.

Chapter 3: RealOne Player Events

Refer to this chapter for information about the Javascript and ActiveX event handlers that provide information about RealOne Player activity.

Part II: Scripting with the Embedded Player

The chapters in this section explain how to use Javascript and VBScript in the embedded environment, in which RealOne Player functions as a browser helper application, but does not launch as a separate application.

Chapter 4: Embedded Environment

This chapter provides an overview of the embedded player, with descriptions of the `<EMBED>` and `<OBJECT>` tags used to embed presentations.

Chapter 5: Embedded Method Overviews

Read this chapter for an overview of the scripting methods available when you embed a presentation in a Web page.

Chapter 6: Embedded Player Methods

This chapter provides complete descriptions of all the embedded methods you can use to control an embedded presentation.

Chapter 7: Embedded Player Callbacks

This chapter lists all the callback methods that return information about the embedded presentation.

Conventions Used in this Guide

The following table explains the typographical conventions used in this guide.

Notational Conventions

Convention	Meaning
emphasis	Bold text is used for in-line headings, user-interface elements, URLs, and e-mail addresses.
terminology	Italic text is used for technical terms being introduced, and to lend emphasis to generic English words or phrases.
`syntax`	This font is used for fragments or complete lines of programming syntax (markup).
`syntax emphasis`	Bold syntax character formatting is used for program names and to emphasize specific syntax elements.
`variables`	Italic syntax character formatting denotes variables within fragments or complete lines of syntax.
`[options]`	Square brackets indicate values you may or may not need to use. As a rule, when you use these optional values, you do not include the brackets themselves.
`choice 1|choice 2`	Vertical lines, or "pipes," separate values you can choose between.
…	Ellipses indicate nonessential information omitted from examples.

Additional Documentation Resources

In addition to this guide, you may need the following resources, which are available for download at **http://service.real.com/help/library/encoders.html**:

- *Helix Producer User's Guide*

  This user's guide gives you the step-by-step instructions for running Helix Producer™, which turns audio and video files into streaming RealAudio® and RealVideo® clips. An online version of this guide is available through the Helix Producer **Help** menu.

- *RealNetworks Production Guide*

  This guide is the main reference manual for streaming media production. Refer to it for instructions and tips on media production, as well as for

complete information about using SMIL 2.0, RealText, and other clip types.

- *Introduction to Streaming Media with RealOne Player*

  This guide provides a simple, streamlined introduction to RealOne Player production techniques. For new users, it provides basic information about streaming media, clip-encoded URLs, Ram files, and SMIL 2.0.

Technical Support

To reach RealNetworks' Technical Support, please fill out the form at:

- **http://forms.real.com/service/techsupport/contact.html**

The information you provide in this form will help Technical Support personnel respond promptly.

RealForum

RealNetworks also encourages you to join RealForum, an e-mail discussion group about RealNetworks 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**.

PART

I

SCRIPTING WITH REALONE PLAYER

The following chapters explain how to use Javascript and ActiveX controls when creating presentations that play in the RealOne Player native interface.

CHAPTER

1

REALONE PLAYER ENVIRONMENT

This chapter introduces you to the RealOne Player media and browsing environment, explaining the various authoring techniques you can use to create compelling media presentations. It also covers the functional areas in which you can use Javascript or ActiveX to create innovative Web presentations.

The Three-Pane Environment

RealOne Player integrates streaming media with HTML pages simply and effectively. Because previous versions of RealPlayer did not natively display HTML pages, linked pages opened in the viewer's default Web browser, which split the presentation between separate applications. RealOne Player closes this divide, benefitting both the viewer, who does not have to switch between applications to watch an integrated presentation, and the presentation author, who can more easily coordinate streaming media with Web pages.

As with past RealPlayers, you can still embed streaming media in any Web page that viewers display in their favorite Web browsers. Although embedding is a widely used means of integrating streaming media with HTML content, the required embedding markup can be cumbersome. With RealOne Player, you can keep your streaming media and HTML pages separate, coordinating the two with simple production techniques. This reduces the work required to stream media and display HTML pages simultaneously.

The following figure illustrates the three-pane environment of RealOne Player, which is based on the metaphor of "play/more/explore." Here, the Media Playback pane plays streamed or downloaded clips. The Related Info pane gives the viewer more information about the presentation. And the detachable Media Browser pane lets the viewer explore the World Wide Web. This design gives you one pane for playing media, one pane for displaying small HTML pages related to the media, and one pane for showing large Web pages, such as your home page.

RealOne Player Three-Pane Environment with a Secondary Browsing Window

For More Information: The presentation planning chapter of *RealNetworks Production Guide* contains a more in-depth overview of the RealOne Player panes.

The Media Playback Pane

The media playback pane hosts media clips and includes buttons for play, pause, rewind, volume control, and so on. Any streaming or downloaded media playable in RealOne Player can display in this pane. This includes the core RealOne Player clip types and markup languages:

- RealAudio® for audio
- RealVideo® for video
- RealText® for timed text

- RealPix™ for still-image slideshows
- Macromedia Flash for animation
- SMIL for creating an integrated presentation from multiple clips

In addition, RealOne Player can play many other media types, including MPEG audio and video.

The Related Info Pane

The related info pane, which is also called the "context pane," appears to the right of the media playback pane. It's designed to display small HTML pages that supplement media clips. These pages might contain album cover art, copyright information, advertisements, and so on. Although using the related info pane is not required, displaying supplemental HTML pages in this pane greatly enhances the viewing experience. The related info pane can display any HTML page content supported by Microsoft Internet Explorer version 4.

Because the media playback and related info panes are separate, you can easily open multiple HTML pages as a presentation plays, displaying each page at a specific point in the media timeline. You can thereby update the related info pane simply by opening a new HTML page. RealOne Player thereby lets you focus on your media, and display any number of supplemental HTML pages by using simple production techniques.

The Media Browser Pane

The media browser pane can attach to, or detach from, the media playback pane and related info pane. When attached, it appears below the two other panes. Detached, it appears as a stand-alone window that the viewer can close independently of the media playback and related info panes. Sending an HTML page URL to a closed media browser pane reopens the pane, however.

Through the media browser pane, RealOne Player users can surf the Web, play CDs, access their personal media libraries, transfer clips to portable players, and so on. Presentation authors can also use this pane to display Web pages associated with a streaming presentation. The pane can display any content supported in Microsoft Internet Explorer version 4, including Javascript. You might use this pane to display your home page after a media presentation plays, for example.

Now Playing List

In the left side of the media browser pane, viewers can display a clickable "Now Playing" list. When the viewer plays a streaming media clip or presentation, the clip or presentation title displays in this list. Additionally, the viewer can build a clip list by dragging media links from an HTML page displayed in the related info or media browser pane.

RealOne Player 'Now Playing' List

Secondary Browsing Windows

Like most Web browsers, RealOne Player can display any number of additional browsing windows, which are always detached from the three-pane environment. You can display Web pages associated with your presentation in secondary browsing windows, for example. Displaying full Web pages in the media browser pane is preferable in most cases, though, because many viewers are likely to have that pane already attached to the media playback and related info panes. Additionally, only the media browser pane includes the "Now Playing" list.

Controlling Interactions Between RealOne Player Panes

RealOne Player supports a variety of authoring languages and techniques that allow content in one pane to control content in another pane. The following sections describe these languages and techniques, helping you to decide how to create a presentation based on how you want the presentation to function.

Displaying HTML Pages Through Streaming Media

With your streaming media, you can use three techniques to open URLs in the related info or media browser pane. Although these techniques do not involve scripting, they are compatible with the scripting methods covered later in this

guide. They allow you to create "media-driven" presentations, in which supplemental information displays in the HTML panes based on the media timeline, or viewer interaction with media clips.

> **Note:** Although RealOne Player can play proprietary formats used by other media players, such as Windows Media and QuickTime, it does not support the use of a Ram file or SMIL with these formats. When streaming one of these formats to RealOne Player, you must author presentations using the markup conventions supported by Windows Media Player or QuickTime Player, respectively.

Appending HTML URLs to Media URLs in a Ram File

You typically launch media clips that play in RealOne Player with a Ram file, which uses the extension `.ram`. The plain text Ram file, which you can link to any Web page with a standard `<a href>` tag, launches RealOne Player, and gives it the full HTTP or RTSP URL to a media clip or SMIL presentation. Within the Ram file, you can append URLs that open HTML pages in the related info or media browser pane. This Ram file method is easy to use, and is well-suited for simple presentations, such as a single RealVideo clip that displays an HTML page as it plays.

> **For More Information:** For full information about the Ram file syntax, see the presentation delivery chapter of the *RealNetworks Production Guide*. *Introduction to Streaming Media with RealOne Player* also covers this topic in its Ram file chapter.

Embedding HTML URLs In a RealVideo or RealAudio Clip

When you create a RealVideo or RealAudio clip with Helix Producer, you can write an *events file* that defines one or more URLs that open in a RealOne Player HTML pane at certain points as the clip plays. You then use a utility that embeds the events file into the clip. Whenever you stream the clip, the encoded URLs open automatically. This technique works only with RealAudio and RealVideo clips. Because it encodes URLs directly into the clip, it is not recommended if you want the HTML pages associated with clips to change over time, or you want to stream the clip without opening the URLs.

For More Information: For information about merging an events file with a clip through the **rmevents.exe** utility, see *Introduction to Streaming Media with RealOne Player*.

Using SMIL to Display HTML Pages

To lay out and synchronize multiple media clips, you use Synchronized Multimedia Integration Language (SMIL) to create simple to highly complex media presentations. A SMIL presentation always plays in the media playback pane, but it can also open HTML pages in the other panes. Using SMIL gives you far more control over HTML display than using a Ram file or encoding URLs directly into clips. The following are some of the capabilities that SMIL gives you in the RealOne Player environment:

- Open any number of HTML pages automatically at any point.
- Open HTML pages interactively, such as when the viewer clicks a graphic image displayed in the media playback pane.
- Use a powerful timing model to define exactly when HTML pages open. For example, SMIL lets you open a page whenever a clip finishes playing. You do not need to know how long the clip lasts.
- Manage bandwidth for complex presentations. SMIL lets you "prefetch" clip and HTML page data before the pages display.
- Display different HTML pages based on the viewer's preferred language, available bandwidth, monitor color depth, or many other criteria.

Note: RealOne Player supports both SMIL 1.0 and SMIL 2.0. Only SMIL 2.0 lets you open URLs automatically in the RealOne Player HTML panes, though.

For More Information: For full information about SMIL and RealOne Player, see *RealNetworks Production Guide*. For a basic introduction to SMIL, see *Introduction to Streaming Media with RealOne Player*.

Controlling Content Through the HTML Panes

Through HTML pages displaying in the related info or media browser, you can control the media displaying in the media playback pane, as well as open new HTML pages. These methods, which you can mix with the media-based techniques described above, allow you to create "user-driven" presentations, in

which clips and supplemental information display according to viewer interaction within the HTML panes.

Opening URLs with Simple Links

Because the related info or media browser pane display any HTML content, the most basic way to control the presentation is to add simple hypertext links in the form `<a href>` to the HTML pages that display within these windows:

- A simple HTML hypertext link can open a new page in the media browser pane from the related info pane. You simply need to add the correct target attribute to the `<a href>` tag:

  ```
  <a href="URL" target="_rpbrowser">
  ```

  Any other target name will open the HTML page in a secondary window that is detached from the basic three-pane environment. You should not attempt to open an HTML page in the related info pane with a simple link in the media browser pane, however, because the related info pane URL requires sizing information that you cannot pass in the link. The Javascript/ActiveX methods let you pass this information, though.

- If you link to a Ram file with a simple `<a href>` link, the clip or SMIL presentation given in the Ram file URL automatically plays in the media playback pane. You do not need to use any additional pane targeting method.

  For More Information: For full information about the Ram file syntax, see the presentation delivery chapter of the *RealNetworks Production Guide*. *Introduction to Streaming Media with RealOne Player* also covers this topic in its Ram file chapter.

  Tip: To avoid a file downloading dialog, you can use the Javascript/ActiveX methods for playing clips when the viewer clicks certain links.

Javascript and ActiveX Methods

RealOne Player supports several methods that work with both Javascript and ActiveX. These give you the most control over the presentation through the HTML pages displaying in the related info or media browser pane. The remainder of this chapter describes how to use these methods.

Using Javascript and ActiveX in the RealOne Player Environment

The Javascript and ActiveX methods available in the RealOne environment are superior to simple hypertext links for opening media in the media playback pane, or for displaying HTML pages. Additionally, these methods let you build an interactive application that lets viewers perform functions such as adding clips to the "Now Playing" list, adding clips to the favorites list, and displaying RealOne Player dialog boxes.

> **Note:** These extensions are customized for the RealOne Player media environment, and will not work in external Web pages. In addition, the environment itself depends upon the client-installed version of Internet Explorer. RealOne Player requires Internet Explorer v4.0 or later to run optimally.

The RealOne Player environment can be accessed from within the player itself, or from Web pages external to the player. Use the Javascript methods and events to perform functions from within the player environment. Use the RealOne Player ActiveX control to provide content to the RealOne Player environment from Web pages external to the player.

> **Tip:** To see sample files, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 2, and view this page.

Using Javascript Methods and Events

To use the available Javascript methods and events, you must declare them in the script section of a Web page displayed in the media browser pane or the related info pane.

When using the Javascript methods and events from within the RealOne Player environment, all methods are in the "external" container of the document object model (DOM), that is, they are appended to `window.parent.external`. Therefore, a call to `PlayClip` in RealOne Player would look like the following:

```
window.parent.external.PlayClip(...)
```

Using ActiveX Controls

The ActiveX controls provided with RealOne Player are similar in scope to the Javascript methods and events, with some limitations, such as no ability to

preload URLs. However, these controls let you target and, to a great degree, control RealOne Player from an external browser as you would from within the RealOne Player environment. To use the ActiveX control, declare it in the body of your Web page with the following class ID:

```
CLSID:FDC7A535-4070-4B92-A0EA-D9994BCC0DC5
```

for example:

```
<OBJECT ID="RealOneActiveXObject" WIDTH=0 HEIGHT=0
CLASSID="CLSID:FDC7A535-4070-4B92-A0EA-D9994BCC0DC5">
</OBJECT>
```

In script code, you can then call methods on this object using its object ID:

```
RealOneActiveXObject.PlayClip(...)
```

Using RealOne Player Methods

The RealOne Player methods provide a means of directly accessing the player and modifying content through your web page. These methods are broken down into basically two categories, those that customize playback and dynamically open URLs, and those that retrieve information.

Customizing Playback and Dynamically Opening URLs

The RealOne Player environment can synchronize the display of associated URLs with a playing clip. Several methods allow you to play a clip in the media browser pane and simultaneously display a URL in the related info pane when playback begins. In addition, you can specify the height and width of the related info pane as you want it to appear when the clip plays.

Specifying a URL for the related info pane is optional in all of these methods. If you do not specify a URL with the related info pane as the target, the specified clip will play in the media playback pane and the related info pane will not open. If a URL is specified without a target listed, it will automatically display in the media browser pane.

Because the related info pane is cleared each time a new presentation begins, the related info pane is automatically closed by default if you play a new presentation without specifying a new related info pane.

Methods for Custom Playback and Opening URLs

Method	Description	Reference
AddToNowPlaying	Adds a clip URL to the clip list.	page 33
ClearNowPlaying	Clears the current playlist.	page 34
HandleAction	Performs a specified action.	page 37
OpenURLInPlayerBrowser	Opens a URL in the media browser pane.	page 40
PlayClip	Sends a URL to the media playback pane.	page 40
PreLoadURL	Called before playback begins.	page 43
SetVideoBackgroundColor	Sets the video background color to the specified value.	page 44

Playing a Clip

Because of the diversity of the RealOne Player environment, there are numerous ways of playing a clip. Clips can be played from an existing web page, or you can devise elaborate combinations of URLs and related information that can be displayed while the clip plays.

The simplest means of playing a clip in the RealOne Player enviroment is to add a single Javascript line to a web page that is to be run in the player's media browser pane or related info pane. This line automatically begins playing a clip you have selected. For example, the following line would play the welcome.rm file:

```
window.parent.external.PlayClip("rtsp://helixserver.example.com/welcome.rm")
```

> **Note:** The zipped HTML+Javascript version of this guide, which you can download from **http://service.real.com/help/library/encoders.html**, contains several sample files. Because some of these files use absolute, local paths, you must copy the entire samples directory to your C: drive (c:\samples) before playing them.

You can also open the RealOne Player and play a clip from an external web page. To do this, you must use an ActiveX control that provides a class identifier that lets you target and control the RealOne Player (using an ActiveX control is described in "Using ActiveX Controls" on page 16). For

example, the following line in an external web page would play the welcome.rm file in RealOne Player:

```
RealOneActiveXObject.PlayClip("rtsp://helixserver.example.com/welcome.rm")
```

The PlayClip method also contains many options for customizing your presentation in the RealOne Player environment. Each of these options can be used indivdually, or in any combination, whatever is required to enhance your presentation.

While your clip is playing, the status display at the top of the RealOne Player contains information provided with each clip that is played. This information can also be obtained using the GetClipInfo method. In some cases, you might want to replace this material with alternative information. The PlayClip method contains a set of optional parameters that modify the information used in the clip. For example, the following code would change the title and artist name:

```
// Play a clip and show new status display
function clipPlay() {
   window.parent.external.PlayClip(
      "rtsp://helixserver.example.com/welcome.rm",
      "Title=Glorious Day|Artist name=Me Alone")
}
```

> **Note:** For more information on using the GetClipInfo method to display all of the information associated with a clip, see "Displaying Clip Information" on page 27

RealOne Player is more than just a player. The RealOne Player environment can also display information related to the clip being played, or let you browse a web page in the media browser. All of these capabilities can be controlled while playing a clip. The PlayClip method includes parameters to add content to the related info pane, which appears beside the current playing clip, and browsing capabilitis in the media browser pane. Both of these capabilities can be used individually, or all at once. For example, the following code displays information in the related info pane, and displays content from a URL in the media browser pane:

```
// Play a clip, display related info, and load URL in media browser
function clipPlay() {
   window.parent.external.PlayClip(
      "rtsp://helixserver.example.com/welcome.rm","",
      "http://www.example.com/welcome.htm", 300, 220,
      "http://www.real.com", "_rpbrowser")
}
```

The size of the related info pane is automatically set by the size of the media clip being played in the media playback pane. However, you can alter the size of the related info pane (along with the height of the media playback pane) by setting the `width` and `height` parameters in the `PlayClip` method, as was done in the previous example.

The previous example also displays the `www.real.com` web site in the media browser pane. If you would prefer to open a separate window for this URL, replace the `_rpbrowser` string in the previous example with any other target name, and the URL will be displayed in a secondary window.

Once you have opened a URL in the related info pane, you can continue to use the information supplied by that URL during playback of other clips. A reserved value, `_keep`, preserves the last URL loaded in the related info pane without having to reload the URL when another clip begins playing.

```
// Play a clip and keep previous related info URL
function clipPlay() {
   window.parent.external.PlayClip(
      "rtsp://helixserver.example.com/welcome.rm","",
      "_keep", 300, 220)
}
```

The next sample shows how to play a clip in the media playback pane and display a URL in the media browser pane, but because no related information is supplied, no related info pane is displayed:

```
// Play a clip and load URL in media browser
function clipPlay() {
   window.parent.external.PlayClip(
      "rtsp://helixserver.example.com/welcome.rm","",
      "", 0, 0, "http://www.real.com", "_rpbrowser")
}
```

The RealOne Player media browser pane contains a "Now Playing" list on the left side. When you play a clip using the PlayClip method, you can specify whether the clip is added to the "Now Playing" list. By default, the clip is

added. To prevent the clip from being added, set the bnow_playing parameter to false, as shown in the following example:

```
// Play a clip, don't add to Now Playing list
function clipPlay() {
   window.parent.external.PlayClip(
      "rtsp://helixserver.example.com/welcome.rm","",
      "", 0, 0, "", "", false)
}
```

The PlayClip method contains one required parameter and seven optional parameters. You do not need to type out all of the optional parameters when you use this method. However, if any optional parameter is used, all optional parameters up to the one being used must contain an entry. For a string parameter, this entry must be a pair of empty quotes. For the integer width and height parameters, you must enter a zero (0) when they are not used.

Using the Now Playing List

The Now Playing list, located on the left side of the media browser pane, shows what clips are currently queued to play and what clips have most recently been played. The AddToNowPlaying method opens the Now Playing list (if it is currently closed) and adds a clip to the play list. You can add multiple clips to the play list using AddToNowPlaying as many times as required since this method does not initiate playback. The only required parameter is the URL for the clip to be added.

This method also contains several optional parameters for loading URLs and displaying clip information. These optional parameters work in the same way as the optional parameters in the PlayClip method. For more information on these optional parameters, see "Playing a Clip" on page 18.

It is a good idea to clear the Now Playing list of any clips that may already be listed before adding any new clips. That way you can create a list that does not contain any clips that may have prevously been included in the clip list. To clear the clip list in the Now Playing list, use the ClearNowPlaying method. This method removes all of the clips currently listed in the Now Playing list.

```
parent.window.external.ClearNowPlaying()
```

Opening a URL in the Media Browser Pane

The PlayClip method lets you open a URL in the media browser pane whenever you start playing a clip in the media playback pane. However, you might want to open a URL in the media browser pane before or after the clip begins

playing. To do this, use the OpenURLInPlayerBrowser method. For example, the following code demonstrates how to open a URL in the media browser pane five seconds after a clip has begun playing:

```
// Open URL five seconds into playback
function RPOnPositionLengthChange(position, length)
{
   if ( position == 5000) {
      parent.window.external.OpenURLInPlayerBrowser("http://www.real.com")
   }
}
```

Caching URLs to Enhance Playback Performance

RealOne Player provides an event handler (RPOnPreLoad) and method (PreloadURL) to cache URLs before a presentation begins playing. While not required, caching URLs locally improves playback quality because RealOne Player does not have to sacrifice bandwidth to retrieve the URLs from a remote location during playback.

The PreloadURL method can be used at any time (even outside of an RPOnPreload event) to load a URL in the browser's cache. However, care should be taken when using this outside of the RPOnPreload event in that referencing a Web page during content playback could appreciably slow the bandwidth from the media stream. You could, for example, use the GetPlayerState method to determine if the user is currently playing a clip and, if not, begin preloading more URLs.

> **Note:** You should test your presentation when preloading numerous URLs to ensure that playback bandwidth is not adversely affected.

The following example shows one possible method of using the PreloadURL method to load URLs in the browser cache before media playback begins:

```
// Preload web pages before media playback begins.
function RPOnPreload()
{
      parent.window.external.PreloadURL("http://server/slide1.html")
      parent.window.external.PreloadURL("http://server/slide2.html")
      parent.window.external.PreloadURL("http://server/slide3.html")
}
```

Handling Actions

The RealOne Player environment includes a method that handles a variety of actions. The `HandleAction` method provides a means of moving the user about in the RealOne Player environment, opening various dialogs, and navigating to specific URLs in the media browser pane or a secondary window.

During your presentation, you might want to move the user around various tabs in the media browser pane, or open the "Now Playing" list to display the clips that are set to play. The `HandleAction` method can open the Web, My Library, CD, Devices, and Radio panes to display whatever information the user requires. In addition, the `HandleAction` method can open the "Now Playing" list to display the current clip list. For example, the following line opens the My Library tab in the media browser pane:

```
parent.window.external.HandleAction("MyLibrary")
```

In some cases, you may need the display the player's Preferences or Equalizer dialogs. The `HandleAction` method can open either of these dialog boxes. In addition, you can open the Preferences dialog box to any individual category or page. For example, the following line opens the Preferences dialog box in the Playback Settings page of the Connection category:

```
parent.window.external.HandleAction("ShowPreferences(Connection,Playback Settings)")
```

The `HandleAction` method also provides a means of navigating to a specified URL, which is displayed in either the media browser pane, or a secondary pane. For example, the following line opens a URL in a secondary window:

```
parent.window.external.HandleAction("NavigateToURL(www.real.com, '_rpexternal')")
```

You can also use the `HandleAction` method to show or hide the artist information in the current clip. For example, the following line hides the artist information in the current clip:

```
parent.window.external.HandleAction("ShowArtistInfo(0)")
```

Setting the Background Color

By default, the background color for the media playback pane is set to black. If the media playback pane size is changed so that it is larger than the media playing in the pane, the background color shows around the edges of the media.

To change the background color, use the `SetVideoBackgroundColor` method. This method takes a string value consisting of one of two formats. The first format type consists of a sting containing the RGB hexadecimal value of the

color you want to use in the form `#RRGGBB`, where `RR` represents the red value, `GG` represents the green value, and `BB` represents the blue value. Each of the color values can be set from 00 to FF in hexadecimal. For example, to set the background color to red:

```
parent.window.external.SetVideoBackgroundColor("#FF0000")
```

The second format type consists of a string containing the red, green, and blue values in an array of the form `rgb(x,x,x)`, where *x* is a decimal value from 0 to 255. For example, to set the background color to blue:

```
parent.window.external.SetVideoBackgroundColor("rgb(0,0,255)")
```

Retrieving RealOne Player Information

The following table summarizes the methods that you can use to retrieve RealOne Player information.

Methods for Retrieving RealOne Player Information

Method	Description	Reference
`ComponentVersion`	Retrieves the version of an updated component.	page 35
`GetClipInfo`	Returns desired clip information as a string.	page 36
`GetPlayerState`	Returns the player's current state.	page 37
`InstalledComponents`	Retrieves a list of DLLs that have been installed by RealOne Player.	page 39
`PlayerProperty`	Retrieves the value of a specified RealOne Player property, such as the language preference or the operating system name.	page 42
`RealPlayerVersion`	Detects the RealOne Player version and returns it in packed form.	page 43

Retrieving Version Information

If you are preparing new content for your web page, it might be necessary to determine the version of the user's player and its components before attempting to run updated material. RealOne Player contains two methods that retrieve information about the player version and about individual components of the player.

The `RealPlayerVersion` method returns the version information about the user's installed player as an integer value in a packed format. You can use this information to determine if the user is running a version of the player that is

capable of playing back your web content. If not, you can either run only those parts of your content that the user's player is capable of running, or you can advise the user to update their player. For example, the following code retrieves the RealOne Player version number and prints it on the browser:

```
// Get RealOne Player version
function playerVersion() {
   var vers=window.parent.external.RealPlayerVersion()
   document.write("The version of RealPlayer is " + vers)
}
```

> **Note:** Currently, the `RealPlayerVersion` method must be used without parenthesis at the end of the method name.

Sometimes it may be necessary to get the version information for an individual component in the player. If an individual codec has been updated, for instance to a new build your code requires, but the version of RealOne Player remains the same, use the `ComponentVersion` method to get the version information for the individual component. This method takes as an argument the individual codec version for which you are trying to retrieve information. For example, the following code demonstrates how to retrieve the version information for the Flash 6.0 codec:

```
// Get packed component version number
function CompVersion() {
   var vers=RealOneActiveXObject.ComponentVersion("Flash:6.0")
   document.write("The Flash plugin version is " + vers)
}
```

The version information returned by `ComponentVersion` and `RealPlayerVersion` is an integer value in a packed format. To unpack the information into a readable format, use the `UnpackVersionNumber` function discussed in "Unpacking Version Information" on page 25.

Unpacking Version Information

Both the `ComponentVersion` and `RealPlayerVersion` methods return version information as an integer value in a packed format. This integer value is useful if you are going to compare other version information against the value returned by these methods.

The version information can also be viewed as an unpacked string. This unpacked string provides the version information in a readable format composed of the major version, minor version, and build number, with each

number separated by a period. This format is consistant with the player's version information as presented in the About RealOne Player dialog box. For example:

```
6.0.10.290
```

The Javascript version of the UnpackVersionNumber function listed below takes the integer value returned by either ComponentVersion or RealPlayerVersion and unpacks it into a readable string. The entire UnpackVersionNumber function must be used as shown below in any page or script to invoke it properly.

```
function UnpackVersionNumber(n)
 {
  return "" + (n >> 28) + "." +
  ((n & 0xFF00000) >> 20) + "." +
  ((n & 0xFF000) >> 12) + "." +
  ( n & 0xFFF);
 }
```

You can also use the UnpackVersionNumber function in VBScript with the following code:

```
Function UnpackVersionNumber(n)
   UnpackVersionNumber = (((n And &HF0000000) / &H10000000) & "." & _
      ((n And &HFF00000) / &H100000) & "." & _
      ((n And &HFF000) / &H1000) & "." & _
      (n And &HFFF))
End Function
```

The following Javascript example demonstrates how to use the UnpackVersionNumber function. The first part of the example declares the function as part of the page, though this could be done in an external script as well. The second part invokes the function on a version number retrieved using RealPlayerVersion, then assigns the resulting string to the variable szPlayerVersion.

```
// Unpack the version number
function UnpackVersionNumber(n)
{
    return "" + (n >> 28) + "." + ((n & 0xFF00000) >> 20) + "." +
        ((n & 0xFF000) >> 12) + "." + ( n & 0xFFF);
}
    ...
{
```

```
    var nRPVersion = window.parent.external.RealPlayerVersion
{
    szPlayerVersion= UnpackVersionNumber(nRPVersion)
}
```

Getting Player Information

When the RealOne Player is installed on a user's system, it saves a set of properties that describe the user's operating system, and information about the player that was installed. You can retrieve this information using the `PlayerProperty` method. The `PlayerProperty` method takes a single parameter that describes the type of property for which you want information returned. For example, the following code returns the user's operating system, the version of the player that is installed, and the bandwidth setting chosen by the user:

```
// Get player property information
function getProperty() {
   document.write("Operating system: " +
      parent.write.external.PlayerProperty("OSNAME") + "<BR>")
   document.write("Product version: " +
      parent.write.external.PlayerProperty("PRODUCTVERSION") + "<BR>")
   document.write("Player bandwidth: " +
      parent.write.external.PlayerProperty("BANDWIDTH"))
}
```

Displaying Clip Information

RealOne Player contains two methods that can retrieve information about the clip in the clip in the media playback pane. The first, `GetClipInfo`, returns information about the clip. The second, `GetPlayerState`, returns the current state of the clip.

Clips loaded in the media playback pane usually contain a set of information that describes the composition of the clip. Many clips, for instance, contain the album name and the name of the artist. Other possible information contained in the clip could be the genre of the clip, the language in which the clip is recorded, the year the clip was recorded, and so on. Use the `GetClipInfo` method to retrieve this information. The `GetClipInfo` method takes as an argument the type of information you want returned, such as the author name, and returns that information from the clip. For example, the following code returns the album name and the name of the artist:

```
// Get the album name and artist
function getInfo() {
   document.write("Album name: " +
      parent.window.external.GetClipInfo("Album name")
   document.write("Artist name: " +
      parent.window.external.GetClipInfo("Artist name")
}
```

Once a clip is loaded in the media playback pane, you can retrieve the current state of the clip. Once you get the state of the clip, you can use this information to determine what actions you take. If the user is listening to the clip, for instance, you could choose to add a new clip to the Now Playing list. Use the GetPlayerState method to retrieve the state of the clip. The GetPlayerState method can determine if the clip is stopped, contacting a URL, buffering, playing, pausing, or seeking in the clip. For example, the following code detects a change in the state of the playback, and displayes the new state:

```
// Get the new state
function getPlayerState() {
   document.write("The player is currently " +
      parent.window.external.GetPlayerState( ))
}
```

Determining Installed Player Components

Before you begin a presentation on RealOne Player, you should check to ensure the player contains all of the components required to display your work. The InstalledComponents method can be used to retrieve the list of DLLs that are currently installed on the user's player. This list can then be compared to the required components for your presentation.

The following example shows how to check the player's installed components. The information returned by InstalledComponents includes the name of each component, along with the version number of that component. The component name and version number are separated by a colon (:), and each component name and version number pair are separated by a pipe symbol (|). This example takes the returned list of components and uses the pipe symbol to parse each component to a separate line.

```
// Parse the installed components
function installComp(){
      var install=RealOneActiveXObject.InstalledComponents()
      document.write("The following components are currently installed:<BR><BR>")
      while(install.indexOf("|")>0){
            component = install.substring(0,install.indexOf("|"));
```

```
            document.write(component + "<BR>");
            install = install.substring(install.indexOf("|") + 1, install.length);
        };
        document.write(install)
}
```

Note: Currently, this method must be used without parenthesis at the end of the method name.

Using RealOne Player Event Handlers

RealOne Player reports the events that occur within the application using a set of predefined functions known as *event handlers*. You can use the event handlers presented here on your Web page to intercept and interpret RealOne Player events, such as capturing user interactions with the application controls, or monitoring the progress of your presentation. The following table summarizes the available events.

Events for Retrieving RealOne Player Information

Method or Event Name	Description	Reference
`RPOnBuffering`	Called when RealOne Player buffers a clip.	page 45
`RPOnPositionLengthChange`	Called at regular intervals as the clip position changes.	page 45
`RPOnPreLoad`	Called before playback begins.	page 46
`RPOnStateChange`	Called when the RealOne Player state changes.	page 46

Handling Media Clip Buffering

The `RPOnBuffering` event handler returns information about the buffering event while the media clip is buffering. In addition, it returns a value that represents the percentage of buffering that has completed.

The following example displays in the current pane any buffering state that is occurring, along with the percentage of buffering that has completed:

```
// Display the current buffering state and percentage
function RPOnBuffering(flags, percent)
{
   buffer = "Buffering: " + flags + ", " + percent + "%<BR>"
   document.write(buffer)
}
```

Determining the Current Time Position

Whenever a media clip is playing in the media playback pane, the position of the clip is automatically updated every half second. You can use this information to schedule events at a specific time during the clip, such as loading URLs in the related info pane or the media browser pane.

The `RPOnPositionLengthChange` event handler returns the current time position within the clip during playback, along with the total duration of the clip. The current position is returned in as the number of milliseconds in half second increments (that is 500 for the first half second, 1000 as the second half second, and so on).

The following example displays a URL in the current pane after the clip has played for five seconds:

```
// Synchronize HTML with your presentation
function RPOnPositionLengthChange(position, length)
{
   if ( postion == 5000 ) {
      flag = 1
      document.location = "http://www.real.com"
   }
}
```

Performing Tasks Before Playing a Clip

Before you begin playing a clip, you can perform tasks that might affect the bandwidth of the clip if the task was performed during playback. This is especially important for users with a low-bandwidth connection, such as a 56K modem.

The `RPOnPreload` event handler processes events that occur before a clip is played. For example, you could use this time to preload URLs destined for the related info pane or the media browser pane. Preloading URLs allows for faster display later on during media playback, especially when synchronizing URLs with the media being played.

An RPOnPreload event occurs whenever the player encounters a ?rpcontexturl query string in a .RAM, .RM, or .SMI file. In general, the first HTML related info URL should preload all the related info URLs used in the presentation. You should test your presentation, however, to ensure the number of related info URLs you are preloading does not delay the beginning of the playback too long, and that the presentation works as expected.

The following example preloads two URLs that will be used later during playback:

```
// Preload the required URLs
function RPOnPreload()
{
   window.parent.PreloadURL("http://www.real.com")
   window.parent.PreloadURL("http://service.real.com")
}
```

Detecting a State Change

Sometimes it might be necessary to determine the current state of the player. For example, you might want to know if the user is currently playing back a clip and, if so, you can add a clip to the Now Playing list rather than using the PlayClip method, which would be more intrusive

The RPOnStateChange event handler returns the state of the clip being played whenever the state changes. The value returned by this event handler is an integer that indicates whether the clip has stopped, has started playing, has been paused, and so on.

The following sample performs a specific task depending on the state that is returned:

```
// Display the current state of the clip
function RPOnStateChange(newstate)
{
   var statestring=""
   switch(newstate) {
      case 0:
         statestring += "Stopped"
         break
      case 1:
         statestring += "Contacting"
         break
      case 2:
         statestring += "Buffering"
```

```
            break
         case 3:
            statestring += "Playing"
            break
         case 4:
            statestring += "Paused"
            break
         case 5:
            statestring += "Seeking"
            break
         default:
            break
      }
      document.write("The player is currently " + statestring)
   }
```

CHAPTER 2

REALONE PLAYER METHODS

This chapter provides a alphabetized reference to the methods you can use in the RealOne Player environment. For an overview of the methods, see "Using Javascript and ActiveX in the RealOne Player Environment" on page 16.

AddToNowPlaying

Opens the "Now Playing" list and adds a URL to the clip list after the current clip. Optionally, it displays an associated URL in the related info pane with the specified height and width when the added clip plays, and a URL to the media browser pane. Available as a Javascript extension and an ActiveX control.

```
AddToNowPlaying(url, clipinfo, related_info_url, width, height, media_browser_url, target)
```

url
: String containing the URL to add to the the RealOne Player clip list. This parameter is required.

clipinfo
: String of extra clip information, such as title, author, and so on. This parameter contains a string of name-value pairs, `<keyword=value>`, separated by pipes. This parameter is optional. Valid keywords for `clipinfo` are:

- Album name
- Artist name
- CDNum
- Comments
- Genre
- Language

- Mood
- Preference
- Situation
- Title
- Year

Example:

```
"Title=XXX|Artist name=XXX|Album name=XXX|Genre=XXX;"
```

> **Note:** Each keyword/value pair is separated by the pipe symbol (|). For each entry to work correctly, you must ensure there are no spaces before or after each pipe symbol.

related_info_url
: String containing the URL to display in the related info pane when the added clip plays. This parameter is optional.

width
: Integer that specifies the width of the related info pane in pixels. This parameter is not required, but its use is recommended. If no width is specified, the width defaults to 320 pixels

height
: Integer that specifies the height of the related info pane in pixels. This parameter is not required, but its use is recommended. If no height is specified, RealOne Player uses the height of the media presentation.

media_browser_url
: String containing a URL to display in the URL designated by the target parameter. This parameter is optional.

target
: Optional string indicating the pane in which to open the URL given in the media_browser_url parameter. If this string is set to _rpbrowser, the URL is opened in the media browser pane. Any other target name displays the URL in a secondary browsing window.

ClearNowPlaying

Clears the RealOne Player's current playlist and stops any clips currently playing. You should call this method before using any of the other methods to

synchronize playback. Available as a Javascript extension and an ActiveX control.

```
ClearNowPlaying()
```

Returns `void`.

> **Note:** Currently, this method must be used without parentheses at the end of the method name.

ComponentVersion

Returns the version of an updated component in packed form. Available as a Javascript extension and an ActiveX control.

```
ComponentVersion( name )
```

name

The name of the component to be examined. This parameter must be specified in the following format:

```
[name]:[major version].[minor version]
```

The following table identifies the possible values for component names. The values are case-sensitive and must be entered exactly as they appear below.

Component Names

Name	Component
audp	Extra audio plugin
DBCMpg1	MPEG video plugin
Flash	Macromedia Flash plugin
GF	GIF plugin (for backwards compatibility)
GFJP	JPG plugin (for backwards compatibility)
imgp	All image plugin (gif, jpg, png)
MP3PL	MP3 Playlist plugin
MPGA	MP3 Audio plugin
PNG	PNG plugin (for backwards compatibility)
RA	RealAudio
RealTxt	RealText
RichFX	RichFX plugin

(Table Page 1 of 2)

Component Names (continued)

Name	Component
RPix	RealPix
RV	RealVideo
sdp	Scalable Multi-cast plugin
vidp	Extra video plugin

(Table Page 2 of 2)

Note: This list is not exhaustive because components are being added to RealOne Player all the time. This list will be updated as available, but you should contact RealNetworks if you have a special need.

For example, use the following to check the version of the Flash 6.0 plugin:

```
window.parent.external.ComponentVersion('Flash:6.0')
```

Returns a string that contains the version of the component in packed form. This information could then be used to determine if the plugin meets the minimum requirements to play a requested presentation. If not, an upgrade request could be initiated.

To unpack this number into a more readable form, use the `UnpackVersionNumber` function, described in "Unpacking Version Information" on page 25, as follows:

```
UnpackVersionNumber(window.parent.external.ComponentVersion('RA:6.0'))
```

GetClipInfo

Retrieves the specified value of the clip information as authored by the media provider in a string format. This method only works for clips launched from HTML in the related info pane. Available as a Javascript extension.

```
GetClipInfo( property )
```

property
: Specifies the requested information. One of the following:

- Album name
- Artist name
- CDNum

- Comments
- Genre
- Language
- Mood
- Preference
- Situation
- Title
- Year

Returns a string suitable for insertion as part of the HTML in the related info pane.

GetPlayerState

Gets the player's current state. Available as a Javascript extension.

```
GetPlayerState( )
```

Returns an integer that describes the player's current state. One of the following:

Value	Description
0	Indicates the player is currently stopped.
1	Indicates the player is currently contacting.
2	Indicates the player is currently buffering.
3	Indicates the player is currently playing.
4	Indicates the player is currently paused.
5	Indicates the player is currently seeking.
6	Indicates the player is busy showing a modal dialog box.

HandleAction

Performs a specified action. Available as a Javascript extension and an ActiveX control.

```
HandleAction( action )
```

action

The specific action to perform. One of the following:

Action	Performs the Following
`CD`	Opens the CD tab in the player.
`MyDevices`	Opens the Devices tab in the player.
`MyLibrary`	Opens the My Library tab in the player.
`NavigateToURL( url, target )`	Opens the specified URL in the specified target pane. The target pane can be defined as either `_rpbrowser` or `_rpexternal`.
`NowPlaying`	Opens the "Now Playing" list.
`Radio`	Opens the Radio tab in the player.
`ShowArtistInfo ( show )`	Shows or hides the artist information for the current clip. If `show` is set to `0`, the artist information is hidden. If `show` is set to `1`, the artist information is shown.
`ShowEqualizer`	Brings up the player's Equalizer dialog box.
`ShowPreferences( category, page )`	Brings up the player's Preferences dialog box to the selected category and page. Category and page combinations are described below.
`Web`	Opens the Web tab in the player.

The `ShowPreferences` action can take one of the following category and page pairs:

Category, Page	Performs the Following
`General,General`	Opens the Preferences dialog box to the General category.
`Connection,Connection`	Opens the Preferences dialog box to the main Connection category.
`Connection,Playback Settings`	Opens the Preferences dialog box to the Playback Settings page of the Connection category.
`Connection,Internet Settings`	Opens the Preferences dialog box to the Internet Settings page of the Connection category.
`Connection,Proxy`	Opens the Preferences dialog box to the Proxy page of the Connection category.

(Table Page 1 of 2)

Category, Page	Performs the Following
`Connection,Network Transports`	Opens the Preferences dialog box to the Network Transports page of the Connection category.
`My Library,My Library`	Opens the Preferences dialog box to the My Library category.
`My Library,Advanced My Library`	Opens the Preferences dialog box to the Advanced My Library page of the My Library category.
`CD,CD`	Opens the Preferences dialog box to the CD category.
`CD,Advanced CD`	Opens the Preferences dialog box to the Advanced CD page of the CD category.
`Devices,Devices`	Opens the Preferences dialog box to the Devices category.
`Accessories,Accessories`	Opens the Preferences dialog box to the Accessories category.
`Media Types,Media Types`	Opens the Preferences dialog box to the Media Types category.
`Content,Content`	Opens the Preferences dialog box to the Content category.
`Hardware,Hardware`	Opens the Preferences dialog box to the Hardware category.
`AutoUpdate,AutoUpdate`	Opens the Preferences dialog box to the AutoUpdate category.

(Table Page 2 of 2)

Note: Each of the category and page pairs must be capitalized as shown. In addition, there cannot be a space between the category and the comma, nor between the comma and the page.

InstalledComponents

Returns a list of the DLLs and their associated version numbers installed by RealOne Player. Available as a Javascript extension and an ActiveX control.

```
InstalledComponents()
```

Note: Currently, this method must be used without parenthesis at the end of the method name.

Returns a string containing the all of the DLLs installed by RealOne Player and their associated version numbers. The component type and version number are separated by a colon (:). Each DLL is separated by a pipe symbol (|). The following example demonstrates a possible return value:

```
athdb:7.0.0.231|Update:7.0.0.960|ath:7.0.0.231|RNAdmin:0.1.0.548|MSG:7.0.0.552
|PNCRT:6.0.0.0|RMACore:6.0.9.138|vsrc:6.0.7.2119|DRMLite:6.0.8.1860|Player:6.0.1
0.319|Free:6.0.10.319|RA:6.0.9.145|RV:6.0.9.145|Flash:6.0.8.2144|Embed:6.0.8.14
13|RealTxt:6.0.7.2232|imgp:6.0.7.2225|PNG:6.0.7.2061|GF:6.0.7.2236|RPix:6.0.7.2
217|GFJP:6.0.7.2236
```

OpenURLInPlayerBrowser

Opens the specified URL in the media browser pane. Available as a Javascript extension and an ActiveX control.

```
OpenURLInPlayerBrowser( url )
```

url
: Specifies the URL to open in the media browser pane.

PlayClip

Plays a clip from the specified URL in the media playback pane. Optionally sends a URL to the related info pane with the specified height and width. In addition, a URL can be displayed in the media browser pane or a secondary window. Available as a Javascript extension and an ActiveX control.

```
PlayClip(url, clipinfo, related_info_url, width, height, media_browser_url, target,
bnow_playing)
```

url
: String containing the URL to play in the media playback pane. This parameter is required.

clipinfo
: String of extra clip information, such as title, author, and so on. This parameter is entered as a string of name-value pairs, `<keyword, value>`, separated by pipes. This parameter is optional. Valid keywords for this parameter are:

  - Album name
  - Artist name
  - CDNum

- Comments
- Genre
- Language
- Mood
- Preference
- Situation
- Title
- Year

The following example demonstrates a possible entry for the `clipinfo` parameter:

```
"Title=XXX|Artist name=XXX|Album name=XXX|Genre=XXX;"
```

> **Note:** Each keyword/value pair is separated by the pipe symbol (|). For each entry to work correctly, you must ensure there are no spaces before or after each pipe symbol.

related_info_url
: String containing the URL to display in the related info pane. Use the reserved value "`_keep`" to keep the last web page loaded in the related info pane. This parameter is optional.

> **Note:** RealOne Player caches related info pane URLs associated with a presentation. This cache is flushed when a new presentation begins.

width
: Integer that specifies the width of the related info pane in pixels. This parameter is not required, but its use is recommended. If no width is specified, the width defaults to 320 pixels.

height
: Integer that specifies the height of the related info pane in pixels. This parameter is not required, but its use is recommended. If no height is specified, RealOne Player uses the height of the media presentation.

media_browser_url
: String containing the URL to display in the pane designated by the `target` parameter.

target
: String indicating the pane in which to open the URL given in the media_browser_url parameter. If this string is set to _rpbrowser, the URL is opened in the media browser pane. Any other target name displays the URL in a secondary browsing window.

bnow_playing
: Boolean that specifies if a clip is added to the Now Playing list. If set to true (default), the clip is added to the Now Playing list. If set to false, the clip is not added.

Although you can use an <A HREF> in your HTML to link to a .ram file, you should use the PlayClip method instead. If you use <A HREF>, a browser is first invoked, which in turn invokes RealOne Player. If you use the PlayClip method, RealOne Player is invoked directly, which avoids intermediate dialogs being displayed.

PlayerProperty

Retrieves the value of the specified property. Available as a Javascript extension and an ActiveX control.

```
PlayerPropery( property )
```

property
: A string that specifies the appropriate player property. One of the following:

Property	Performs the Following
APIVERSION	Gets the version of the player's Javascript extensions.
BANDWIDTH	Gets the user's bandwidth setting.
COUNTRYID	Gets the country name.
DISTRIBUTION CODE	Gets the OEM distribution code.
INSTALLPATH	Gets the full path to the player's installation directory.
LANGUAGEID	Gets the language of the installed player.
LANGUAGEPREFERENCE	Gets the user's preferred language.
OSNAME	Gets the computer's operating system name, such as Win98.
PRODUCTNAME	Gets the name of the installed player, such as RealOne Player.

Property	Performs the Following
PRODUCTVERSION	Gets the version of the installed player, for example, 6.0.10.448.
REGIONDATE	Gets the Zip or postal code.

Returns the value of the specified property as either a string or int32 (as appropriate).

PreloadURL

Caches URLs to be used later during playback. Caching these URLs enables a quicker display time during media playback without reducing bandwidth during streaming. Available as a Javascript extension.

```
PreloadURL( url )
```

url
Specifies the URL to cache.

The following example shows how you might cache two different URLs (www.example1.com and www.example2.com) before playback. These URLs could be referenced later using PROnPositionChange or other events.

```
<head>
<script language=Javascript>
   function RPOnPreload(){
      PreloadURL(http://www.example1.com)
      PreloadURL(http://www.example2.com)
      }
</script>
…
```

RealPlayerVersion

Retrieves the version number of the installed RealOne Player in packed format. This number indicates the major and minor version information of RealOne Player. Available as a Javascript extension and an ActiveX control.

```
RealPlayerVersion()
```

> **Note:** Currently, this method must be used without parenthesis at the end of the method name.

Returns an integer containing the packed form of the RealOne Player version information. This information can then be compared against the minimum version number required to view your presentation.

Use the `UnpackVersionNumber` function described in "Unpacking Version Information" on page 25 to display the number as a string. The string returned by `UnpackVersionNumber` is the same as that displayed in the **Help>About RealOne Player** dialog.

SetVideoBackgroundColor

Sets the video background color to the specified value. Available as a Javascript extension.

```
SetVideoBackgroundColor( color )
```

color
: The color value specified in either RGB format, for example `"RGB(r,g,b)"`, or as a hexidecimal value in the format "`#RRGGBB`", such as `"#FFFFFF"`.

Returns `void`.

CHAPTER

3

REALONE PLAYER EVENTS

This chapter provides a alphabetized reference to the event handlers you can use in the RealOne Player environment. For an overview of events and event handlers, see "Using RealOne Player Event Handlers" on page 29.

RPOnBuffering

Indicates the type of buffering currently occurring, and the percentage of the buffering that has completed.

```
window.parent.extrenal.RPOnBuffering( flags, percent_complete )
```

flags
: Integer value indicating the type of buffering. One of the following:

Value	Description
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.

percent_complete
: The amount of buffering completed, in percent.

RPOnPositionLengthChange

Indicates the current time position in the clip, and the total length of the clip. This event is called twice per second (that is, every 0.5 second) during the presentation playback.

```
windows.parent.external.RPOnPositionLenghtChange( position, length )
```

position
: Contains the current position of the clip, in milliseconds.

length
: The total length of the clip, in milliseconds.

The following example uses RPOnPositionLengthChange to synchronize HTML with your presentation when the presentation is running between the 5 and 10 second marks.

```
var flag = 0
 function RPOnPositionLengthChange(position, length)
  {
   if ( position >= 5000  && position < 10000  &&  flag == 0 ) {
   flag = 1
   document.location = "http://www.real.com"
  }
 }
```

RPOnPreload

Performs tasks, such as preloading URLs, before loading the media stream in the Media Playback pane.

```
RPOnPreload()
```

The following example shows how you could call PreloadURL within RPOnPreload to cache two different URLs (www.example1.com and www.example2.com) before playback.

```
<head>
<script language=Javascript>
   function RPOnPreload()
   {
      parent.window.external.PreloadURL(http://www.example1.com)
      parent.window.external.PreloadURL(http://www.example2.com)
   }
</script>
...
```

RPOnStateChange

Indicates the play state of RealOne Player has changed (for example from Play to Pause).

```
window.parent.external.RPOnStateChange( newPlayState )
```

newPlayState

Integer value indicating the current state. One of the following:

Value	Description
0	Indicates the player is currently stopped.
1	Indicates the player is currently contacting.
2	Indicates the player is currently buffering.
3	Indicates the player is currently playing.
4	Indicates the player is currently paused.
5	Indicates the player is currently seeking.
6	Indicates the player is busy.

PART

II

PART II: SCRIPTING WITH THE EMBEDDED PLAYER

The following chapters explain how to use Javascript and ActiveX controls when creating presentations that play within a Web page through the embedded RealPlayer or RealOne Player.

CHAPTER

4

EMBEDDED ENVIRONMENT

By embedding RealPlayer or RealOne Player controls in a Web page, you can incorporate streaming media directly into the page. The embedded player environment lets you place presentations in a Web page using only a simple embedding markup. But you can also extend that functionality through Javascript or VBScript, using methods that set and retrieve presentation attributes, control the clip playback, and handle user interactions.

Understanding Presentation Embedding

The following sections provide an overview to help you decide whether to embed your presentation, and, if so, which markup and scripting languages to use.

Embedded Environment vs. RealOne Environment

Before you begin embedding presentations in Web pages, read Chapter 1 to determine if using the native RealOne Player environment, which requires less work than embedding a presentation, suits your needs better. RealOne Player can natively display HTML pages without the overhead of having to include the embedding markup. You may find that developing presentations in the native RealOne Player environment saves you considerable time and effort.

How Embedding Works

To embed a streaming media presentation in a Web page, you add `<EMBED>` or `<OBJECT>` tags to your Web page markup, depending on whether you want to use the Netscape plug-in method, or the ActiveX embedding method (for more information, see "The Two Embedding Methods" on page 53). These tags allow you to add media windows and RealPlayer controls, such as **Stop** and **Start** buttons, directly to your Web page. Additionally, you can use

Javascript or VBScript to extend the functionality of the embedded components.

The Embedded Player

When a viewer surfs to your Web page, the viewer's browser launches RealOne Player's embedded player (or the embedded player of an earlier RealPlayer if that software is installed on the viewer's machine). The embedded player, which is always installed with RealOne Player, handles media playback as a browser helper application, without launching RealOne Player as a separate application. If the viewer does not have RealOne Player installed, the browser typically prompts the viewer to download and install the application.

> **Tip:** It's also a good idea to include a RealOne Player download icon on your Web page. You can find these icons at **http://www.realnetworks.com/company/logos/index.html**.

Backwards Compatibility

The embedding markup and methods described in this guide are geared for RealOne Player. Most markup and methods are backwards compatible with RealPlayer G2, RealPlayer 7, and RealPlayer 8. Some methods are even compatible with RealPlayer 5. RealOne Player's embedded controls have a different look from earlier players, however. Hence, a **Stop** button embedded in your Web page will look different for a viewer who has RealOne Player installed, than for a viewer who has RealPlayer 8 installed. The buttons will function the same way, though.

> **For More Information:** The section "Embedded Controls" on page 78 illustrates the RealOne Player embedded controls.

SMIL in Embedded Presentations

In a Web page, you can embed a single clip, a sequence of clips, or a SMIL presentation, which can coordinate many clips to a single timeline. You can play an entire SMIL presentation in a single image window in your Web page, or you can display each clip within the presentation in a separate window. RealOne Player supports SMIL 2.0 and 1.0, whereas RealPlayer G2, RealPlayer 7, and RealPlayer 8 support only SMIL 1.0. Therefore, if you embed a SMIL 2.0 presentation, viewers who have RealPlayer G2 through RealPlayer 8 installed are prompted to upgrade to RealOne Player.

Media Preparation

Before you create an embedded presentation, you'll need to create your media. RealOne Player supports a wide range of streaming media clip types, including RealAudio, RealVideo, RealText (streaming, timed text), RealPix (streaming still images), Flash animation, and MPEG video and audio. You'll also need to decide which bandwidth targets you want to meet, whether dial-up modems, fast connections such as cable modems, or both.

> **For More Information:** See *RealNetworks Production Guide* for basic information about clip types and bandwidth management. If you plan to use RealAudio or RealVideo, you'll need Helix Producer, which you can download from **http://www.realnetworks.com/products/producer/index.html**.

The Two Embedding Methods

The embedded player supports two types of markup that allow you to add streaming media to your Web page. The first method uses the Netscape plug-in architecture, which adds `<EMBED>` tags to your Web page and allows you to control playback with Javascript commands. Any browser that supports this plug-in architecture will be able to play your embedded presentation. Major browser support includes the following:

- Netscape Navigator 4.0 and higher.

  There are known compatibility issues with some versions of Netscape Navigator 6.0, although versions 6.1 and 6.2 work properly. However, always be sure to test for compatibility.

- Microsoft Internet Explorer 5.0 and higher.

  Even when you use the `<EMBED>` tag, RealOne Player communicates with the Internet Explorer browser using ActiveX technology. This makes the `<EMBED>` tag compatible with all versions of Internet Explorer, including version 6.

Using the Netscape plug-in method allows you to reach the widest Internet audience. However, the embedded player's ActiveX control lets you use a second method, which adds `<OBJECT>` tags to your Web page and allows the use of VBSript. This method 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++.

Both embedding methods support the same tag parameters. Plus, the Javascript and VBScript methods are virtually identical. Because they both have the same capabilities, you can use either the Netscape plug-in or the ActiveX control depending on which products you need to support, and the audience you wish to reach.

Javascript and VBScript

Once you embed a presentation, you can use a scripting language such as Javascript or VBScript to extend the presentation's functionality. Scripting lets you add functions like stop, play, and volume control to elements such as forms, HTML buttons, or graphic images. For example, you can use your own graphic image for a **Stop** button, capturing mouse clicks to stop the clip playback. While the methods are most commonly accessed from Java, Javascript, or VBScript, they can also be developed using C++ and other programming languages.

> **Tip:** If you decide not to use scripting, you may find the Web page embedding chapter of the *RealNetworks Production Guide* easier to use than this guide. That chapter covers basic embedding without scripting. You can download this guide from **http://service.real.com/help/library/encoders.html**.

Methods

Both the Netscape plug-in and ActiveX embedding methods support the same methods that allow you to issue commands and receive information about the viewer's embedded player, such as its version. Chapter 5 introduces you to these methods according to functional category. Chapter 6 is a reference for all available methods.

Callback Events

In addition to scripting methods that let you issue commands to the embedded player, the player supports *callback* methods that report events. You can use these callback methods to intercept and interpret RealOne Player events. This lets you track mouse movement, capture user interactions with the application controls, and monitor the progress of your presentation, for example. Chapter 7 is a reference for all available callbacks.

Using the Netscape Plug-in

To use the 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 optional parameters are also available. The syntax for a typical <EMBED> tag looks like the following:

```
<EMBED SRC="presentation.rpm" WIDTH=300 HEIGHT=134 PARAMETER=value>
```

The preceding sample tag creates a playback area 300 pixels wide by 134 pixels high within your Web page, and displays the contents of presentation.rpm within the playback area. Typically, your Web page will contain multiple <EMBED> tags, each of which embeds a different RealOne Player control. You link all of these tags together using the CONSOLE parameter.

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

> **For More Information:** For a list of all <EMBED> parameters, see "Tag Parameters" on page 61. The section on the SRC parameter contains information about linking your Netscape plugin to your media. Available controls are described in "Embedded Controls" on page 78.

Extending Embedded Controls 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 as described above. The following example shows an <EMBED> tag with the required SRC, WIDTH, and HEIGHT parameters, as well as several additional parameters described in "Tag Parameters" on page 61:

```
<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 work with RealPlayer, the <EMBED> tag must **not** contain the parameter NOJAVA=true. This parameter prevents the Java Virtual Machine from starting up in Netscape version 4.x, but it has no effect on Netscape 6.0 or Internet Explorer browsers.

Once you create your tag or tags, you can use Javascript to issue 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:

```
<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>
```

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.

Receiving Callbacks Through Javascript

RealPlayer communicates the events that occur in a Netscape plug-in through a set of internal callback routines. Depending on the platform and version of Netscape you are targeting, however, the embedded player 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, use LiveConnect to receive the callbacks.

Handling Events in Netscape Navigator 6 or later

Netscape Navigator version 6.0 or later does not support callback event handling in the same manner as previous versions. For this reason, the embedded player build number 6.0.8.1024 (RealPlayer 8 embedded player, update 3) 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 in the configurations listed in the following table. It is not available with C++ or ActiveX.

Supported Configurations for the New Event Handling Mechanism

Browser	Version	Windows	Macintosh	Unix
Netscape Navigator	6.0 or later	yes	yes	no*
Microsoft Internet Explorer	all versions	no	yes	no*

* Unix does not currently support callback event handling.

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 ...>
```

Do not worry about backward compatibility: all versions of the embedded player ignore unrecognized tags. However, you may detect the player version, and, if it is earlier than build 6.0.8.1024, inform the user that their RealPlayer version does not support the new event handling features.

> **For More Information:** The GetVersionInfo method detects the player version. Version numbers are described in "Obtaining RealOne Player Version Information" on page 96.

Sending Callbacks to Multiple Plug-Ins with Netscape Navigator 6 or Later

If you embed multiple plug-ins in the same page, you need to identify which plug-in receives which callback. To do this, include a NAME parameter (described on page 69) with a unique, user-defined value in each <EMBED> tag:

```
<embed NAME=RVOCX SCRIPTCALLBACKS=All ...>
```

Then, prepend each callback with the NAME value, separating the NAME value and callback method with an underscore:

```
function RVOCX_onClipOpened(short_clip_name,url){ ... }
```

The NAME value must also appear in embedded methods:

```
<a href="#" onClick="document.embeds.RVOCX.DoPlay()">Play</a>
```

Handling Events in Netscape Navigator 4.x

When developing for Netscape versions 4.x, you must use LiveConnect to receive the callbacks sent by RealOne Player. LiveConnect is described in Netscape documentation available at the following Web address:

http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/

To receive callbacks, you must embed a Java <APPLET> tag in your HTML code. This tag should include a reference to an event interface class file (for example, CODE="callback.class"), and have the MAYSCRIPT attribute set. In addition, you must also provide a NAME attribute (such as NAME="*MyName*") to identify the applet instance, as shown here:

```
<APPLET CODE="callback.class" WIDTH=1 HEIGHT=1 NAME="MyName" MAYSCRIPT>
</APPLET>
```

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

```
if(document.MyName.OnClipClosed())
```

The HTML+Javascript version of this guide contains a rudimentary callback Java applet in the samples directory (callback.class and callback.java) for testing the callback methods of the embedded player. You can use this applet to exercise your callback routines, or modify the callback.java file and compile your own class file to more fully meet the needs of your application.

Class Files

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 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.

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.

Using the ActiveX Control

To use the embedded 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"
```

It must also specify the width and height of the playback area. When you intend to use scripting with the control, you must also give a user-defined 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. Typically, your Web page will contain multiple <OBJECT> tags, each of which embeds a different RealOne Player control. You link all of these tags together using the CONSOLE parameter.

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 letter case, although this manual depicts them in uppercase. Except for file names, which are typically 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 "Tag Parameters" on page 61. The section on the SRC parameter contains information about linking your ActiveX control to

your media. Available controls are described in "Embedded Controls" on page 78.

Extending Embedded Controls 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://helixserver.example.com/video1.rm">
<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 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:

```
<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>
```

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.example.com/video1.rm">
</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>
```

Tag Parameters

This section describes, in alphabetical order, the parameters that you can add to an <EMBED> or <OBJECT> tag. In an <EMBED> tag, parameters take the following form:

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

In an <OBJECT> tag, parameters take this form:

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

AUTOGOTOURL

Specifies how to handle URLs embedded in a presentation.

Value(s):	true|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, RealOne Player sends the URLs to a Java

applet or other application through the OnGotoURL callback. If you do not include this parameter in your tag definition, and your presentation contains embedded URLs, the URLs are not passed to the browser.

> **Tip:** Beginning with RealPlayer G2, you can also use the SetAutoGoToURL method described on page 118 to dynamically change how embedded URLs are handled at any time.

AUTOSTART

Specifies whether to automatically play a presentation

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

When set to true, AUTOSTART starts the clip playing as soon as the clip's preroll has been streamed. The viewer does not need to press a **Play** button. If multiple controls are linked together with the CONSOLE parameter, AUTOSTART needs to be set in just one control. With the default value of false, the presentation, the viewer must press a **Play** button to start the presentation.

> **Tip:** If you are developing a Netscape plug-in in RealPlayer version 5.0 or later, you can also use the SetAutoStart method described on page 119 to dynamically control automatic playback at any time.

BACKGROUNDCOLOR

Sets the background color for the image window.

Value(s):	*color_name*|*#RRGGBB*
Default Value:	black
Compatibility:	RealPlayer G2 or later

The background color is specified using an RGB hexadecimal color value (#RRGGBB) or a color name. When a clip includes transparent regions, the background color shows through these areas. If you do not include the BACKGROUNDCOLOR parameter in your tag definition, the background color for

the image window is set to black (default). The following table lists the valid background color values:

white (#FFFFFF)	silver (#C0C0C0)	gray (#808080)	black (#000000)
yellow (#FFFF00)	fuchsia (#FF00FF)	red (#FF0000)	maroon (#800000)
lime (#00FF00)	olive (#808000)	green (#008000)	purple (#800080)
aqua (#00FFFF)	teal (#008080)	blue (#0000FF)	navy (#000080)

> **Tip:** You can also use the SetBackgroundColor method described on page 119 to dynamically change the background color of the image window at any time.

CENTER

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

Value(s):	true|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 omitted or set to false, the presentation is not centered and the presentation expands to fill the image window.

> **Tip:** You can also use the SetCenter method described on page 120, to dynamically center the presentation in the image window at any time.

> **Warning!** The CENTER and MAINTAINASPECT parameters cannot both be set to true. In addition, the set methods for these parameters (SetCenter and SetMaintainAspect) cannot also 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 player 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 the ActiveX Control" on page 59.

CONSOLE

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

Value(s):	*name*|_master| _unique
Default Value:	_unique
Compatibility:	RealPlayer 5 or later

When there are multiple controls on the same page, a shared console name links the controls into 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 presentation. The following valid console names are valid:

name	A user-defined name that links the control with other controls that share the same name. For example: console1
_master	A predefined name that links the control to all other controls in the page.
_unique	A predefined name that does not link the control to any other controls on the page.

> **Tip:** You can also use the SetConsole method described on page 121 to dynamically specify whether your controls are linked at any time.

You can have multiple console names for separate presentations. For a page showing two video clips, for example, you can define the console names video1 and video2. All controls linked by video1 interoperate, as do all controls linked by video2. But a video1 volume slider, for example, will not affect the volume of a video2 clip.

Tips for Using Consoles

- Every <EMBED> tag must have a SRC attribute. Tags linked by a console name should have the same SRC value.
- If the <EMBED> tags in a console group have different SRC values, the first valid source that RealOne Player finds among those choices becomes the console source. This may not always be the first source listed.
- Clicking a play button for one console stops playback for other consoles. This allows multiple consoles to play separate audio tracks or to use the same image window.

Multiple Controls Example

The following example sets up an image window and two sets of controls (a play button and stop button) for two separate videos, video1.rm and video2.rm. The predefined console name _master links the image window to both control sets. The control sets use different console names, however, so they do not link to each other. Clicking each play button therefore starts a different video.

Because each <EMBED> tag must have a SRC value, the image window in the following example uses the same source as the first play button. The viewer simply clicks either play button to start a video. Clicking the other play button stops the first video and plays the second one:

```
<EMBED SRC="video1.rpm" CONSOLE=_master WIDTH=176 HEIGHT=128
NOJAVA=true CONTROLS=ImageWindow>

<H4>Video 1</H4>
<EMBED SRC="video1.rpm" CONSOLE=video1 WIDTH=44 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton>
<EMBED SRC="video1.rpm" CONSOLE=video1 WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=StopButton>
```

```
<H4>Video 2</H4>
<EMBED SRC="video2.rpm" CONSOLE=video2 WIDTH=44 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton>
<EMBED SRC="video2.rpm" CONSOLE=video2 WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=StopButton>
```

CONTROLS

Embeds the specified RealPlayer control on your Web page.

Value(s):	*control_name*
Default Value:	All
Compatibility:	RealPlayer 5 or later

The embedded player includes many controls that you can add to your Web page or application, including an image window, a full control panel, individual buttons, sliders, and status panels. For a complete listing of controls, see "Embedded Controls" on page 78.

> **Tip:** You can also use the SetControls method described on page 122 to dynamically add controls to your Web page at any time.

HEIGHT

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

Value(s):	*pixels*|*percentage*
Default Value:	(none)
Compatibility:	RealPlayer 5 or later

This parameter sets the height of the control in pixels, or as a percentage of the displayed browser window. 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.

> **Note:** All embedded controls have a recommended width and height. For a complete listing of controls, see "Embedded Controls" on page 78.

ID

Identifies the embedded presentation for reference by VBScript.

Value(s):	*unique_ID*
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 you have identified your presentation in an `<OBJECT>` tag:

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

you can 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 the ActiveX Control" on page 59.

LOOP

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

Value(s):	true|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` needs to be be set to `true` in only one tag definition.

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.

> **Tip:** You can also use the SetLoop method described on page 123 to dynamically change whether the clip should loop at any time.

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|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 needs to be set to true in only one tag definition.

If the MAINTAINASPECT parameter is set to false, or if it is not included in the tag definition, the clip scales as necessary to allow it to fill the image window completely. 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.

> **Tip:** You can also use the SetMaintainAspect method described on page 124 to dynamically change whether the correct aspect ratio should be maintained at any time.

NAME

Specifies the name to associate with an embedded RealPlayer control, to enable Javascript to refer to the control.

Value(s):	*name*
Default Value:	(none)
Compatibility:	Netscape only, RealPlayer 5 or later

To refer to an embedded 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>
```

A Javascript command can then refer to the control like this:

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

When using more than one instance of a single type of embedded control, or a variety of different embedded controls in your Web page, each instance must have a unique NAME value. 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 refer to named controls only 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. Control referencing from Javascript may therefore be unavailable.

NOJAVA

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

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

Setting NOJAVA=true in *every* tag linked by the same console name 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, *if* 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.

NUMLOOP

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

Value(s):	*integer*
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. When you have multiple, linked controls, you need to set NUMLOOP in one tag only. 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.

> **Note:** 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 or method is ignored. This condition still applies even if NUMLOOP has been set to zero.

> **Tip:** You can also use the SetNumLoop method described on page 124 to dynamically specify the number of times the presentation should loop at any time.

PARAM

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

Value(s):	any valid parameter, except NAME
Default Value:	(none)
Compatibility:	ActiveX only, RealPlayer 5 or later

Additional parameters are specified through 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, which is described on page 59.) 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 the ActiveX Control" on page 59.

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|false
Default Value:	false
Compatibility:	RealPlayer 5 or later

By setting PREFETCH to true, you can obtain information about a presentation before it begins playing, and use that information to change playback characteristics. When an embedded player detects that prefetch playback mode is enabled, it obtains the the presentation's stream description information. After the information has been captured, OnPreFetchComplete (see page 140) is returned to the plug-in or control, and the presentation is paused.

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 104). 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.

> **Tip:** You can also use the SetPreFetch method described on page 125 to dynamically specify whether prefetch playback mode is enabled at any time.

> **Note:** SMIL 2.0 includes a <prefetch/> tag, which is unrelated to this PREFETCH parameter. The SMIL tag lets you download all or part of a clip's *stream data* before the clip plays, whereas the PREFETCH parameter obtains just the *stream description*. For more on SMIL prefetching, see the *RealNetworks Production Guide*.

REGION

Defines the ImageWindow control in which a specific clip from a SMIL presentation plays.

Value(s):	*SMIL_region_name*
Default Value:	(none)
Compatibility:	RealPlayer G2 or later

This parameter is for use only when you embed a SMIL presentation, *and* you want to play clips from the presentation in separate image windows on your Web page. This parameter is not necessary if you want to embed an entire SMIL presentation in a single image window. In this case, you just treat the SMIL presentation like a clip, embedding it in a single image window set to the same size (or larger) as the SMIL presentation's width and height, which is set in the SMIL file's <root-layout/> tag.

Clips in a SMIL presentation normally play in SMIL regions that are defined within the SMIL file header. When you embed the presentation's clips in different image windows, though, you omit the SMIL layout information. But you still include in the SMIL source tag for each visual clip a region attribute that ostensibly assigns the clip to a SMIL region. For example, a RealText clip included in a SMIL presentation might have a source tag that looks like this:

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

To embed newsarticle.rt in a specific image window, you create the image window with CONTROLS=ImageWindow, and use the REGION parameter to identify the playback region. Here is an example using the Netscape plugin:

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

The next example is for the ActiveX control:

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

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

> **For More Information:** For more information on SMIL, see the latest *RealNetworks Production Guide*. That guide's Web page embedding chapter contains more information on this topic, and includes sample files of embedded SMIL presentations.

SCRIPTCALLBACKS

Specifies the callback events to handle in a comma-separated list.

Value(s):	*callback_name*|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 through a comma-separated list:

```
SCRIPTCALLBACKS=OnPresentationOpened,OnPresentationClosed
```

You can include any of the callbacks described in Chapter 7 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 Navigator 6 or later" on page 56.

SHUFFLE

Specifies whether all unplayed clips in a presentation, should be played back in a random order

Value(s):	true|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. This parameter can be used with multiclip RAM files (.ram or .rpm), or with SMIL files that contain only a sequence of clips. 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 file.

> **Tip:** You can also use the SetShuffle method described on page 127 to dynamically specify whether clip playback should be randomized at any time.

SRC

Specifies the URL of the .rpm file or 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. Any directories or files specified in the URL cannot contain spaces in their names, that is, they need to be properly URL encoded. For example, use %20 for a space character.

Using the TYPE Parameter

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 this may produce unexpected results. Therefore, it is strongly recommended that you always include the SRC parameter and, minimally, supply the name of an empty presentation file.

> **Tip:** You can also use the SetSource method described on page 127 to dynamically specify the URL of the presentation at any time.

Specifying a Source With the Netscape Plugin

You must include the SRC parameter in every <EMBED> tag, even when the tag embeds a RealOne Player control, such as a **Play** button, instead of a clip. However, you don't specify a clip or SMIL file directly with SRC. Instead, you specify a Ram file that has a .rpm extension:

```
<EMBED SRC="http://www.example.com/presentation.rpm" WIDTH=176 HEIGHT=132>
```

The .rpm extension causes the browser to use RealOne Player as a helper application, rather than to launch it as a separate application. The .rpm file is a simple text file that gives the full URL to your clip or SMIL file:

```
rtsp://helixserver.example.com/video1.rm
```

> **For More Information:** For full information about the Ram file syntax, see the presentation delivery chapter of the *RealNetworks Production Guide*.

Developing Your Presentation

The easiest means for developing your embedded presentation is to keep your clips in the same folder as your Web page on your desktop computer. Your <EMBED> tag can then link to a .rpm file in that folder:

```
<EMBED SRC="presentation.rpm" WIDTH=300 HEIGHT=134>
```

To embed a single video, for example, the .rpm file simply contains a local file URL to the clip (the file:// protocol designation is required):

```
file://video.rm
```

> **Warning!** For embedded playback to work with Netscape Navigator 6, the path to the .rpm file on a server or your local computer cannot contain spaces or even escape codes for spaces (%20). This causes Navigator 6 to search for a missing plug-in.

Delivering Your Presentation

When you are ready to deliver your presentation to your audience, move your files to their respective servers and change the URLs in your files:

- Keeping the .rpm File and the Web Page Together

  If you plan to keep the .rpm file with the Web page, you do not need to change the SRC values in your <EMBED> tags. You can simply transfer your .rpm file and your Web page to the same directory on your Web server.

- Putting the .rpm File and the Web Page in Different Locations

  If you move the .rpm file to a different directory than that Web page, link each <EMBED> tag's SRC parameter to the .rpm file with a full HTTP URL:

  ```
  SRC="http://www.example.com/media/presentation.rpm"
  ```

- Linking to Streaming Clips

  No matter where you put your .rpm file and your clips, modify the .rpm file to give the fully-qualified URL to the embedded clip or SMIL file. If the clip or SMIL file is on a Web server, use an HTTP URL. If the clip or SMIL file is on Helix Server, use an RTSP URL.

  > **Tip:** Always use a full URL in the .rpm file, even if all files and clips are in the same directory on a Web server. RealOne Player uses the .rpm file to locate the clip or presentation. Without a fully-qualified URL, RealOne Player must construct the location from the original Web page URL and the information in the .rpm file. This creates more possibility for errors.
  >
  > **For More Information:** For more information about the RTSP protocol, as well as the Helix Server Ramgen feature, which lets you eliminate the .rpm file, see the presentation delivery chapter of *RealNetworks Production Guide*.

- Linking to Local Clips

  If you will make your presentation available to people on their local machines (through a download or a CD, for instance), you do not need to change any URLs from those described in "Developing Your Presentation" on page 75. In rare cases, though, you may want to use an absolute link, rather than a relative link, in the .rpm file. When using absolute links, use forward slashes in paths to create "Web style" paths. For example, instead of this URL:

```
file://c:\media\presentation.rpm
```

use this URL, which includes three forward slashes in file:///, and uses forward slashes in path names as well:

```
file:///c:/media/presentation.rpm
```

Specifying a Source with ActiveX

For the ActiveX control, the <OBJECT> tag's CLASSID parameter causes the presentation to play in the Web page, so you can simply link to the SMIL file or clip within a single <OBJECT> tag. It is not necessary to use a Ram file with the .rpm extension:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"...>
<PARAM NAME="SRC" VALUE="rtsp://helixserver.example.com/video1.rm">
...
</OBJECT>
```

TYPE

Identifies the MIME type of the presentation specifed in the SRC parameter, which is described on page 74.

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://.../.../presentation.rpm" WIDTH=176 HEIGHT=132
TYPE="audio/x-pn-realaudio-plugin">
```

The browser first reads the MIME type value, 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.

> **Note:** 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, minimally, supply the name of an empty presentation file.

WIDTH

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

Value(s):	*pixels*|*percentage*
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.

> **Note:** All embedded controls have a recommended width and height. For a complete listing of controls, see "Embedded Controls" on page 78.

Embedded Controls

With the CONTROLS parameter, you can add controls such as a play/pause button to your Web page. Viewers can then control playback as if they were using RealOne Player as a separate application. For example, the following tag displays the play/pause button in your Web page using an <EMBED> tag:

```
<EMBED SRC="presentation.rpm" WIDTH=26 HEIGHT=26 NOJAVA=true
CONTROLS=PlayButton CONSOLE="one">
```

For the ActiveX control, the <OBJECT> tag would look like this:

```
<OBJECT ID=RVOCX CLASSID="clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA"
WIDTH=26 HEIGHT=26>
<PARAM NAME="CONTROLS" VALUE="PlayButton">
<PARAM NAME="CONSOLE" VALUE="one">
</OBJECT>
```

> **Note:** When adding more than one control to your Web page, link the controls together with the CONSOLE parameter.

The following sections list and describe the embedded controls in alphabetical order by control name. You use an <EMBED> or <OBJECT> tag's WIDTH and HEIGHT parameters to set the control's size. Specifying different pixel sizes other than the suggested values scales the controls larger or smaller. You can also use percentage values for sizes, but this is recommended only for the image window.

Tip: Unless noted otherwise, all the RealOne Player controls listed below are compatible with RealPlayer G2, RealPlayer 7, and RealPlayer 8. With those versions of RealPlayer, however, the controls take on a different appearance.

All

The `CONTROLS=All` parameter displays the basic RealOne Player control panel. The control name "default" also works. Functions include play/pause, stop, fast-forward, and rewind. Sliders include a position slider and a volume slider with a mute button that pops up when the speaker button is clicked. Below the buttons are a clip information field, a status panel, a network congestion indicator, and a clip timing field.

Suggested pixel width: 375
Suggested pixel height: 100

If you set the size of this control panel to less than the recommended width or height, the panel drops certain controls instead of squeezing all of the controls down to a smaller size. This lets you add the control panel to small pop-up windows, for example, without the controls becoming difficult to use. This works for RealOne Player, but not earlier versions of RealPlayer.

Width less than 336 pixels:	network congestion indicator dropped
Width less than 306 pixels:	clip timing field dropped
Width less than 226 pixels:	`Clip Info` label, rewind button, and fast-forward button dropped
Width less than 174 pixels:	RealOne logo dropped
Height less than 81 pixels:	clip information field dropped

ControlPanel

Use `CONTROLS=ControlPanel` to display a compact RealOne Player control panel. Functions include play/pause, stop, fast-forward and rewind. There's also a position slider, along with a volume slider and mute button that pops up when the speaker button is clicked.

Suggested pixel width: 350

Suggested pixel height: 36

If you set the size of this control to less than the recommended width, the panel drops certain buttons instead of squeezing all of the buttons down to a smaller size. This works for RealOne Player, but not earlier versions of RealPlayer.

Width less than 220 pixels:	rewind and fast-forward buttons dropped
Width less than 168 pixels:	RealOne logo dropped

FFCtrl

The `CONTROLS=FFCtrl` parameter displays a fast-forward button.

Suggested pixel width: 26

Suggested pixel height: 26

HomeCtrl

The `CONTROLS=HomeCtrl` parameter displays the RealOne Player logo, which is linked to the RealNetworks Web site. In earlier versions of RealPlayer, this control displays the Real™ logo.

Suggested pixel width: 30

Suggested pixel height: 30

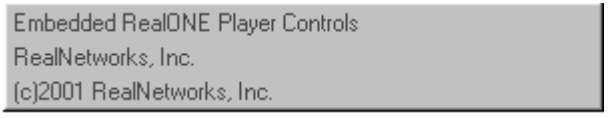
ImageWindow

The `CONTROLS=ImageWindow` parameter displays a playback window. This control is not required for audio-only presentations. Even if no other controls are visible on the page, the user can typically right-click (on Windows) or hold down the mouse button (on the Macintosh) in the playback area to display a menu of choices such as **Play** and **Stop**.

Suggested pixel width: 176 or greater

Suggested pixel height: 132 or greater

InfoPanel



The `CONTROLS=InfoPanel` parameter displays the presentation information panel. For more on presentation information, see the *RealNetworks Production Guide*.

Suggested pixel width: 300

Suggested pixel height: 55

InfoVolumePanel

Use CONTROLS=InfoVolumePanel to display presentation information along with the volume slider and mute button. For more on presentation information, see the *RealNetworks Production Guide*.

Suggested pixel width: 325

Suggested pixel height: 55

MuteCtrl

The CONTROLS=MuteCtrl parameter displays a mute button.

Suggested pixel width: 26

Suggested pixel height: 26

MuteVolume

The CONTROLS=MuteVolume parameter displays a mute button and volume slider.

Suggested pixel width: 26

Suggested pixel height: 88

PauseButton

The CONTROLS=PauseButton parameter displays a pause button. Because the PlayButton control turns into a pause button as a presentation plays, the PauseButton control is generally not necessary with the RealOne Player. To ensure backwards compatibility with earlier versions of RealPlayer, however, use both the PlayOnlyButton and the PauseButton controls.

Suggested pixel width: 26

Suggested pixel height: 26

PlayButton (also PlayOnlyButton)

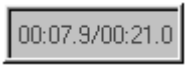
The CONTROLS=PlayButton parameter displays a play button. This turns into a pause button when the presentation plays. If your presentation is accessible to RealPlayers earlier than the RealOne Player, use CONTROLS=PlayOnlyButton instead. In earlier RealPlayers, the PlayButton control includes both play and pause buttons, whereas the PlayOnlyButton control includes just the play

button as shown here. Using `PlayOnlyButton` therefore ensures backwards compatibility.

Suggested pixel width: 36

Suggested pixel height: 26

PositionField



The `CONTROLS=PositionField` parameter displays the position field, which shows the clip's current place in the presentation timeline, along with the total clip length.

Suggested pixel width: 90

Suggested pixel height: 30

PositionSlider

The `CONTROLS=PositionSlider` parameter displays a clip position slider.

Suggested pixel width: 120

Suggested pixel height: 26

RWCtrl

The `CONTROLS=RWCtrl` parameter displays a rewind button.

Suggested pixel width: 26

Suggested pixel height: 26

StatusBar

The `CONTROLS=StatusBar` parameter displays the status panel, which shows informational messages. It also includes the network congestion LED and the

position field, which shows the clip's current place in the presentation timeline, along with the total clip length.

Suggested pixel width: 335

Suggested pixel height: 30

If you set the width of the status bar lower than the recommended width, the panel drops fields instead of squeezing all of the fields down to a smaller size. This works for RealOne Player, but not earlier versions of RealPlayer.

Width less than 330 pixels:	network congestion indicator dropped
Width less than 300 pixels:	clip timing field dropped

Note: The status bar is included in the `All` control. If you do not embed a status bar or status field in your page, error messages display in the browser's status bar.

StatusField

The `CONTROLS=StatusField` parameter displays the message text area of the status bar. If you do not embed a status field or status bar in your page, error messages display in the browser's status bar.

Suggested pixel width: 200

Suggested pixel height: 30

StopButton

The `CONTROLS=StopButton` parameter displays a stop button.

Suggested pixel width: 26

Suggested pixel height: 26

TACCtrl

The CONTROLS=TACCtrl parameter displays an information field. Clip or presentation information scrolls vertically through this field when the clip first plays. The viewer can redisplay this information by clicking the arrow button. Clicking the "i" button displays the full presentation information in a pop-up window. With RealOne Player, if you set the width of the TACCtrl to less than 220 pixels, the Clip Info field is dropped.

Suggested pixel width: 370

Suggested pixel height: 32

For More Information: For instructions on defining clip or presentation information, see the *RealNetworks Production Guide*.

VolumeSlider

The CONTROLS=VolumeSlider parameter displays a volume slider.

Suggested pixel width: 26

Suggested pixel height: 65

CHAPTER

5

EMBEDDED METHOD OVERVIEWS

The embedded RealOne Player contains methods that control the playback of embedded presentations. This chapter introduces these method by category, describing how you can use groups of related methods to enhance embedded presentations.

> **Tip:** To see sample files, get the HTML+Javascript version of this guide as described in "How to Download This Guide to Your Computer" on page 2, and view this page.

Controlling Playback

The following methods let you play, pause, and stop the clip using your own controls, as well as determine a clip's playback status:

- `CanPause`
- `CanPlay`
- `CanStop`
- `DoPause`
- `DoPlay`
- `DoStop`

If you want to supply your own play, pause, and stop controls 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()">
```

These methods can also 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`, the clip is either paused or stopped, and is ready to play. If `CanPlay` returns `false`, the clip is already playing, and is not ready to begin playing again.

Obtaining Play State Information

The following methods get the presentation's current play state, or determine the elapsed and remaining buffering times of a presentation:

- `GetBufferingTimeElapsed`
- `GetBufferingTimeRemaining`
- `GetLastMessage`
- `GetLastStatus`
- `GetPlayState`

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

- 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 appears along the bottom of the image window only 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.

Another possibility is determining 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. 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

Several methods allow you to get or set the current state of the embedded controls on your Web page. They supply information about the current state of the embedded controls, along with a means of dynamically changing the controls' states. You can also use these methods in programming languages such as C++.

The set methods allow you to change the attributes of the embedded player. Many of these methods mirror the the <EMBED> or <OBJECT> tag parameters, and can dynamically change the state of an embedded control. For example, you can use SetBackgroundColor to alter the BACKGROUNDCOLOR parameter of the <EMBED> or <OBJECT> tag, changing the ImageWindow background color dynamically.

The following are the methods associated with parameters (listed in parentheses) used with the <EMBED> or <OBJECT> tag:

- SetAutoGoToURL (AUTOGOTOURL)
- SetAutoStart (AUTOSTART)
- SetBackgroundColor (BACKGROUNDCOLOR)
- SetCanSeek (none)
- SetCenter (CENTER)
- SetConsole (CONSOLE)
- SetControls (CONTROLS)
- SetLoop (LOOP)
- SetMaintainAspect (MAINTAINASPECT)
- SetNumLoop (NUMLOOP)
- SetPreFetch (PREFETCH)
- SetShuffle (SHUFFLE)
- SetSource (SRC)

Additional methods let you get the current state of the attributes of the embedded player. For example, GetBackgroundColor returns a hexadecimal value for the current background color:

- DoGotoURL
- GetAutoGoToURL

- GetAutoStart
- GetBackgroundColor
- GetCanSeek
- GetCenter
- GetConsole
- GetControls
- GetLoop
- GetMaintainAspect
- GetNumLoop
- GetPreFetch
- GetShuffle
- GetSource

Seeking Through a Clip

The embedded player provides several methods that obtain information about the currently-playing clip. This includes determining the present clip position, total length, and whether one can seek through the presentation:

- GetLength
- GetPosition
- SetPosition

During playback, use GetCanSeek to verify whether you can seek through the clip. If the method returns true, you can use GetLength and GetPosition to obtain the clip length (total milliseconds) and current position (milliseconds already played). You can use the value returned from GetLength to ensure that the value used with SetPosition does not exceed the total clip length.

You can also use SetCanSeek when GetCanSeek returns true. This method allows you to specify whether the viewer can seek through the clip. However, if you attempt to use SetCanSeek to allow seeking through an inherently unsearchable clip, such as a live broadcast, the set method has no affect.

Accessing Clip Title, Author, and Copyright Information

The following methods allow you to get or set a clip's title, author, and copyright information:

- GetAuthor
- GetClipHeight

- GetClipWidth
- GetCopyright
- GetTitle
- SetAuthor
- SetCopyright
- SetTitle

When title, author, and copyright information has been encoded in a clip, use GetTitle, GetAuthor, and GetCopyright to retrieve the information. In multiclip presentations, these methods return the information associated with the currently-playing clip.

The SetTitle, SetAuthor, and SetCopyright methods dynamically change whether the title, author, and copyright information displays for a clip. Normally, this information displays when the viewer clicks the "i" button on an embedded control panel. In multi-clip presentations, these methods override the clip information for the entire presentation, not just the currently-playing clip. These methods are useful for multiclip presentations in which you want to supply a title, author, and copyright information for the entire presentation.

Directing a Playlist in a Multi-clip Presentation

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

- DoNextEntry
- DoPrevEntry
- GetCurrentEntry
- GetEntryAbstract
- GetEntryAuthor
- GetEntryCopyright
- GetEntryTitle
- GetNumEntries
- HasNextEntry
- HasPrevEntry

To get the number of entries in the playlist, use GetNumEntries. 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 GetCurrentEntry. Note,

however, that this method begins counting at zero, the third entry in a playlist therefore returns the number 2. While playing a multi-clip presentation, use HasNextEntry to determine if there is another entry in the playlist. You can then use DoNextEntry to jump to the next clip. Use HasPrevEntry to determine if a previous entry exists in the playlist. If so, you can use DoPrevEntry to jump back to the beginning of the previous clip.

After you have determined the number of entries in a playlist, you can use GetEntryAbstract, GetEntryAuthor, GetEntryCopyright, and GetEntryTitle 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

The GetLiveState method indicates whether the current stream is live. This information is useful, for instance, if you want to know whether to use the clip positioning methods. If GetLiveState returns true, the current stream is part of a live presentation, and you cannot use the clip positioning methods.

Display User Interface Dialogs

RealOne Player contains several menus and dialogs that let the viewer set various player preferences, view the playback statistics, and view information about the player. The following methods can set RealOne Player preferences, as well as view player information and statistics:

- GetShowAbout
- GetShowPreferences
- GetShowStatistics
- SetShowAbout
- SetShowPreferences
- SetShowStatistics

You can access preferences using SetShowPreferences. Setting this method's parameter to true brings up an abbreviated version of the RealOne Player Preferences dialog box. To determine if this dialog box is already displayed, use GetShowPreferences, which returns true if the dialog box is already displayed.

Use SetShowStatistics to view playback statistics. Setting this methods's parameter to true brings up the standard RealOne Player statistics dialog box. To determine if this dialog box is already displayed, use GetShowStatistics, which returns true when the statistics dialog box is displayed.

RealOne Player's "about" dialog box contains information about the player version and the individual player components. To view this dialog, set the SetShowAbout parameter to true. To determine if this dialog box is already being displayed, use the GetShowAbout method. If this method returns true, the dialog is already displayed.

Error Handling

You can designate whether or not the RealOne Player embedded in your Web page displays error dialogs by using the following methods:

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

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.

When a player error occurs, the GetLastErrorMoreInfoURL method supplies a URL that can provide more information than is provided by the standard error string. 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 RealOne Player. The values of the error codes and their descriptions can be found in the SDK rmaerror.h file and in the *RealNetworks SDK Developer's Guide*.

Other RealOne Player 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 RealOne Player, 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 RealOne Player, both of these methods return nothing.

Setting the Display Size

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 with the following methods:

- GetFullScreen
- GetOriginalSize
- SetFullScreen
- SetOriginalSize

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.

Controlling Audio

Using the following methods, 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:

- GetMute

- GetStereoState
- GetVolume
- SetMute
- SetVolume

To set the volume of the presentation, use SetVolume. 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 GetVolume 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.

Getting Network Information

Using the following methods, you can obtain information to determine how your embedded presentation reacts to the user's network capabilities.

- GetSourceTransport
- GetPacketsTotal
- GetPacketsMissing
- GetBandwidthAverage
- GetNumSources
- GetBandwidthCurrent
- GetPacketsEarly
- GetPacketsLate
- GetPacketsOutOfOrder
- GetPacketsReceived

Two methods provide playback bandwidth information from the user's RealOne Player. The first, GetBandwidthAverage, returns the average playback

bandwidth of the user's RealOne Player, 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 RealOne Player. `GetPacketsReceived` indicates the total number of packets that the player received correctly from Helix 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 RealOne Player 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 RealOne Player. This method either returns either `local`, `udp`, or `tcp`, depending on the source transport.

Obtaining RealOne Player Version Information

The `GetVersionInfo` method returns the version of the RealOne Player plug-in running on the user's machine (in period-delineated form, such as "6.0.7.788", which is the embedded RealPlayer build for RealPlayer 8). With this information, your Web page can 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 Compatibility Table

Release Version	Standalone Build Number	Embedded Player 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

Note: If you use JavaScript to get the version information, the `GetVersionInfo` may not function on some versions of Internet Explorer 5. In this case, use VBScript to ensure the `GetVersionInfo` method functions properly.

Event Handling

RealOne Player 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:

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

Available Methods

The SetWantKeyboardEvents method sets whether or not keyboard events are returned from the user's RealOne Player. If SetWantKeyboardEvents is set to true, the keyboard event callbacks, OnKeyDown, OnKeyPress, and OnKeyUp, are sent from RealOne Player 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 RealOne Player.

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 RealOne Player 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 RealOne Player.

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.

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. The SetConsoleEvents method does not affect ActiveX controls.

CHAPTER

6

EMBEDDED PLAYER METHODS

An application, applet, or control can use the methods described in this chapter to communicate with the embedded environment of the RealOne Player. The methods are listed here in alphabetical order and each description contains information about how to use the method in your Netscape 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 5.

> **Note:** Many of the methods listed below return a `boolean` value for plug-ins. Plug-in developers can safely ignore the `boolean` value because 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.

CanPause

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

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 RealOne Player 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.

DoNextEntry

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.

DoPause

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

```
DoPause(void)
```

Returns a `boolean` value for plug-ins.

DoPlay

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

```
DoPlay(void)
```

Returns a `boolean` value for plug-ins.

DoPrevEntry

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.

GetAuthor

Indicates the current clip's author string.

```
GetAuthor(void)
```

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

GetAutoGoToURL

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`.

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 RealOne Player preferences.

```
GetConnectionBandwidth(void)
```

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

GetConsole

Indicates a console name used to link multiple control instances.

```
GetConsole(void)
```

Returns a `string` that contains the name of the RealOne Player 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 97 for an explanation of console events.

GetControls

Indicates the name of the visible components of the RealOne Player control.

```
GetControls(void)
```

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

> **For More Information:** For valid control names, see "Embedded Controls" on page 78.

GetCopyright

Indicates the current clip's copyright string.

```
GetCopyright(void)
```

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

GetCurrentEntry

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.

GetDRMInfo

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

```
GetDRMInfo (string identifier)
```

identifier
: A four-letter string identifier for which the license will be returned. For example, the string RNBA is used for the RealNetworks Media Commerce Suite.

Returns a string in the following form (line breaks are for readability only):

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

> **Note:** This method is available only in embedded player 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.

GetEntryAbstract

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

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

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

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.

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 SDK header file `pnresult.h` in the available at:

http://www.realnetworks.com/resources/server/

In normal operation, all components need to be able to handle the following basic codes that may be returned by Helix Server:

- `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.

GetLastErrorSeverity

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. RealOne Player 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. RealOne Player will shut down the presentation if necessary.
2	Critical	Error that may require user intervention to correct. RealOne Player 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.

GetLastMessage

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.

```
GetLastMessage(void)
```

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

GetLastStatus

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 `GetLastMessage` 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.

GetNumEntries

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

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 Helix 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 Helix 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 Helix 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 Helix 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 Helix 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 RealOne Player.

`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`*.

GetPreFetch

Indicates whether or not `PREFETCH` is enabled.

`GetPreFetch(void)`

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

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

Indicates whether or not the RealOne Player Statistics dialog box is visible.

`GetShowStatistics(void)`

Returns `true` if the RealOne Player 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

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 identifies the source protocol used for playback (`local`, `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.

GetVersionInfo

Indicates major and minor version information for the embedded RealOne Player (not the parent RealOne Player).

```
GetVersionInfo(void)
```

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

GetVolume

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.

HasNextEntry

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.

HasPrevEntry

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.

SetAuthor

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

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

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.

```
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.

> **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 62, to specify automatic playback in the tag definition.

SetBackgroundColor

Specifies the desired background color for the image window control.

```
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:

white (#FFFFFF)	silver (#C0C0C0)	gray (#808080)	black (#000000)
yellow (#FFFF00)	fuchsia (#FF00FF)	red (#FF0000)	maroon (#800000)
lime (#00FF00)	olive (#808000)	green (#008000)	purple (#800080)
aqua (#00FFFF)	teal (#008080)	blue (#0000FF)	navy (#000080)

Returns a boolean value for plug-ins.

> **For More Information:** You can also use the BACKGROUNDCOLOR parameter defined on page 62, to specify the background color of the image window in the tag definition.

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.

```
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.

For More Information: You can also use the CENTER parameter defined on page 63, to specify that the presentation should be centered in the image window, in the tag definition.

SetConsole

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.

```
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.

For More Information: You can also use the CONSOLE parameter defined on page 64, to specify whether your controls are linked in the tag definition.

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 97 for an explanation of console events.

SetControls

Sets the visible components of the control.

```
SetControls(string controls)
```

controls
: The name of the RealOne Player 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.

> **For More Information:** You can also use the `CONTROLS` parameter defined on page 66, to add controls to your Web page in the tag definition.

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.

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.

```
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.

```
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.

For More Information: In the tag definition, you can also use the LOOP parameter defined on page 67, to specify whether the clip should loop.

SetMaintainAspect

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

```
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.

For More Information: You can also use the MAINTAINASPECT parameter defined on page 68 to specify whether the correct aspect ratio should be maintained.

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.

SetNumLoop

Sets number of times to loop the clip.

```
SetNumLoop(int32 number_of_loops)
```

number_of_loops
The number of times for the clip to loop.

Returns a boolean value for plug-ins.

> **For More Information:** You can also use the NUMLOOP parameter defined on page 70, to specify the number of times the presentation should loop, in the tag definition.

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.

Returns a boolean value for plug-ins.

For More Information: You can also use the PREFETCH parameter defined on page 71, to specify whether prefetch playback mode is enabled, in the tag definition.

SetShowAbout

Displays the RealOne Player About dialog box.

```
SetShowAbout(boolean set)
```

set
If set to true, the RealOne Player 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

Opens the RealOne Player environment and displays the RealOne Player Preferences dialog box.

```
SetShowPreferences(boolean set)
```

set
If set to true, the RealOne Player 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

Sets the RealOne Player Statistics dialog box to visible.

```
SetShowStatistics(boolean set)
```

set
If set to true, the RealOne Player Statistics dialog box is displayed. If set to false and the RealOne Player 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.

```
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.

> **For More Information:** You can also use the SHUFFLE parameter defined on page 74, to specify whether clip playback should be randomized, in the tag definition.

SetSource

Specifies 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

```
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.

> **For More Information:** You can also use the SRC parameter defined on page 74, to specify the URL of the presentation, in the tag definition.

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.

SetVolume

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.

CHAPTER

7

EMBEDDED PLAYER CALLBACKS

This chapter 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 5.

OnAuthorChange

Sent when the author string changes.

```
OnAuthorChange(string author)
```

author
: The new author string.

Returns `void`.

OnBuffering

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 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

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

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 109 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 SDK. In normal operation, all components need to be able to handle the following basic codes that may be returned by Helix Server:

  - 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 110.

user_string
: The error string from the last error dialog. For more information, see `GetLastErrorUserString` on page 110.

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

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

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 128) 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

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 128) 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

Sent when user releases keyboard key. This callback method is only sent when the set parameter of the SetWantKeyboardEvents (see page 128) 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 when the cursor is placed over the embedded component. This callback method is only sent when the set parameter of the SetWantMouseEvents (see page 129) 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 Flat 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 while the cursor is positioned over the embedded component. This callback method is only sent when the set parameter of the SetWantMouseEvents (see page 129) 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 137.

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 cursor over the embedded component. This callback method is only sent when the set parameter of the SetWantMouseEvents (see page 129) 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 137.

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

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 142 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

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 139 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 139 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 125) 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 while the cursor is positioned over the embedded component. This callback method is only sent when the set parameter of the SetWantMouseEvents (see page 129) 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 the table "Parameter Values for the Possible Mouse Button Events" on page 137.

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 while the cursor is positioned over the embedded component. This callback method is only sent when the set parameter of the SetWantMouseEvents (see page 129) 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 the table "Parameter Values for the Possible Mouse Button Events" on page 137.

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

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 138 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 139.

Returns `void`.

OnTitleChange

Sent when the title string changes.

```
OnTitleChange(string title)
```

title
: The new title string.

Returns `void`.

OnVolumeChange

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`.

GLOSSARY

B

bandwidth
The upper limit on the amount of data, typically expressed as kilobits per second (Kbps), that can pass through a network connection.

bit
The smallest unit of measure of data in a computer. A bit has a binary value, either `0` or `1`.

bit rate
A measure of bandwidth, expressed as the number of bits transmitted per second. A 28.8 Kbps modem, for example, can transmit or receive around 29,000 bits per second.

broadcast
To deliver a presentation, whether live or prerecorded, in which all viewers join the presentation in progress. Contrast to *on-demand*.

buffering
The receiving and storing of data before it is played back. A clip's initial buffering is called *preroll*. After this preroll, excessive buffering may stall the presentation.

byte
A common measurement of data. One byte consists of 8 bits.

C

cable modems
Devices that allow rapid transmission and reception of data over television cable. They are digital devices, unlike dial-up modems, which transmit analog data.

client
A software application that receives data from a server. A Web browser is a client of a Web server. RealOne Player is a client of Helix Server.

clip
A media file within a presentation. Clips typically have an internal timeline, as with RealAudio and RealVideo.

codec
Coder/decoder. Codecs convert data between uncompressed and compressed formats, reducing the bandwidth a clip consumes.

D

download
To send a file over a network with a nonstreaming protocol such as HTTP. Contrast to *stream*.

DSL
Digital Subscriber Line. A technology for transmitting digital data over a regular telephone line at speeds much faster than dial-up modems.

E

encoding
Converting a file into a compressed, streaming format. For example, you can encode WAV files as RealAudio clips.

events file
A text file that specifies URLs to display at specific points as a RealAudio or RealVideo clip plays. A utility included with Helix Producer events file is merged into the clip using a utility.

F

Flash
A software application and an animation format created by Macromedia. RealOne Player can play Flash animations and stream them in parallel with other clips, such as RealAudio clips.

H

Helix Producer
The primary RealNetworks tool for encoding RealAudio and RealVideo clips.

Helix Server
RealNetworks server software used to stream multimedia presentations to RealOne Player.

HTTP
Hypertext Transport Protocol. The protocol used by Web servers to communicate with Web browsers. In contrast, Helix Server streams clips to RealOne Player with RTSP.

K

kilobit (Kb)
A common unit of data measurement equal to 1024 bits. A kilobit is usually referred to in the context of bit rate per unit of time, such as kilobits per second (Kbps).

kilobyte (KB)
A common unit of data measurement equal to 1024 bytes or 8 kilobits.

O

on-demand
A type of streaming in which a clip plays from start to finish when a user clicks a link. Most clips are streamed this way. Contrast to *broadcast*.

P

prefetch
To stream clip data to RealOne Player before the clip plays back. A clip's preroll can be prefetched minutes before the clip plays, for example, masking the preroll from the viewer.

preroll
Buffering that occurs just before a clip plays back. Preroll should be no more than 15 seconds.

presentation
A clip or group of clips streamed from Helix Server to RealOne Player. The presentation can also include HTML URLs that open in the RealOne Player HTML windows.

R

RealAudio
A clip type for streaming audio over a network. RealAudio clips use the `.rm` extension.

RealOne Player
The successor to RealPlayer 8, the RealOne Player combines streaming and digital download technologies. It supports the SMIL 2.0 standard.

RealPix
A clip type (file extension `.rp`) for streaming still images over a network. RealPix uses a markup language for creating special effects such as fades and zooms.

RealPlayer G2
The RealNetworks client software that introduced plug-ins and the ability to update itself. It, along with the later RealPlayer 7 and RealPlayer 8, supports the SMIL 1.0 standard.

RealText
A clip type (file extension .rt) for streaming text over a network. It uses a markup language for formatting text.

RealVideo
A clip type for streaming video over a network. RealVideo clips use the extension `.rm`.

RTP
Real-Time Protocol. The open, standards-based data package protocol Helix Server uses (along with RTSP) to communicate with RTP-based clients. Contrast to *RealAudio*.

RTSP
Real-Time Streaming Protocol. An open, standards-based control protocol that Helix Server uses to stream clips to RealOne Player or any RTP-based client. Contrast to *HTTP*.

S

server
1. A software application, such as a Web server or Helix Server, that sends requested data over a network.

2. A computer that runs server software.

SMIL
Synchronized Multimedia Integration Language. A markup language for specifying how and when each clip plays within a presentation. SMIL files use the extension `.smil`.

stream
1. To send a media clip over a network so that it begins playing back as quickly as possible.

2. A flow of a single type of data, measured in kilobits per second (Kbps). A RealVideo clip's soundtrack is one stream, for example.

SureStream
A RealNetworks technology that enables a RealAudio or RealVideo clip to stream at multiple bit rates.

U

URL
Uniform Resource Locator. A location description that enables a Web browser or RealOne Player to receive a clip stored on a Web server or Helix Server.

V

visualization

An animation built into RealOne Player that the viewer can display when playing audio-only clips.

INDEX