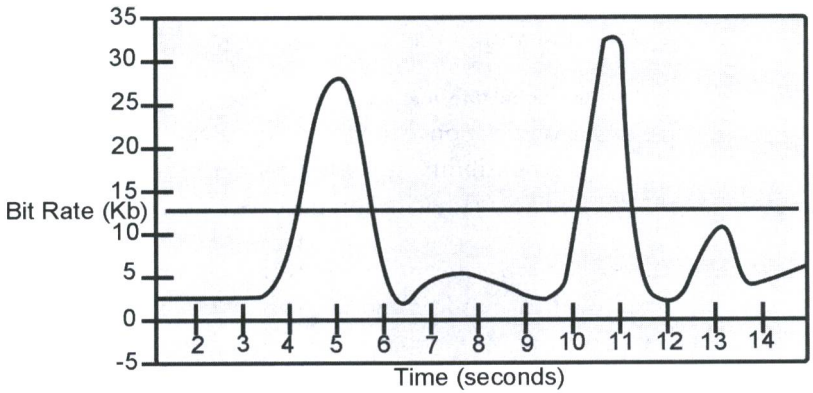
Shockwave Flash Sample Bit Rate Requirement



This graph shows a sample Shockwave Flash clip that targets a bandwidth of 12 Kbps. At five and ten seconds into the clip, the bandwidth requirement spikes because the clip requires more than 12 Kb of data. These spikes typically correspond to scene changes or the introduction of new objects in a key frame. RealPlayer responds by buffering the data as it comes in, potentially delaying playback until all necessary data has arrived.

As you create your Macromedia Flash animation, you need to minimize the spikes that may cause RealPlayer to halt the presentation while it buffers data. There are two ways you can do this:

1. As you create your animation, minimize the overall bit rate requirement of the Shockwave Flash stream by keeping the ratio of file size to clip length as low as possible. This doesn't eliminate spikes, but it helps keep the spikes smaller. The guidelines below explain how to do this.
2. After you export your animation file, use the Shockwave Flash tuning utility to change the streaming file's bandwidth consumption. You can also use the RealFlash Bit Rate Calculation Spreadsheet to view frame-by-frame bandwidth needs.

Additional Information

"Tuning Shockwave Flash" on page 72.

Keeping Shockwave Flash Files Small

The following are recommendations for keeping Shockwave Flash file size down as you develop the animation:

- Reduce key frames.

  Excessive key frame changes increase bandwidth consumption. Minimize the number of key frames and simplify the objects within key frames.

- Use symbols instead of groups.

  Flash stores a symbol once and can refer to it repeatedly, each reference adding little to the file size. However, it stores a group definition each time the group is used. Using a group three times, for example, stores the same data in the file three times. Using symbols instead of groups can therefore reduce file size significantly.

- Minimize morphing.

  The Macromedia Flash 3.0 morphing feature ("shape tweening") creates additional key frames and hence increases the file size and streaming bit rate.

- Simplify elements.

  Simplify the elements drawn in or imported into Macromedia Flash. Under **Modify>Curves**, use the **Smooth** and **Straighten** commands on lines and curves to strip away unneeded point and path information. This reduces the data stored for each element. Use **Optimize** to optimize the data reduction while maintaining acceptable screen appearance. Because screen resolution is lower than print resolution, you can eliminate minute details without compromising appearance.

- Adjust JPEG quality when exporting.

  When exporting .fla files to .swf files, set the JPEG quality to no greater than 50, possibly as low as 30.

Minimizing CPU Usage

Bandwidth is not the only consideration when optimizing files. Macromedia Flash's vector-based animation differs from raster or bitmap animation in that the user's machine must perform complex calculations to display the animation. Operations that require many calculations on top of the normal load may adversely affect playback. The following are ways to reduce RealFlash CPU requirements:

- Set a frame rate of seven (7) fps for 28.8 Kbps connections. This provides acceptably smooth motion without overburdening most processors.

- Optimize tweening.

  Tweening interpolates the motion between key frames. Interpolating multiple objects and color effects at the same time will adversely affect playback. Other actions related to tweening that slow down playback are changing large areas of the screen between frames and using gradient fills.

- Decrease the number and size of objects simultaneously moving on screen.

  RealPlayer must redraw areas where action occurs, thus consuming CPU cycles. To minimize this, localize tweening to a small portion of the screen so that the entire screen does not have to be redrawn. File size remains the same, but only one part of the screen is redrawn.

Using Interactive Commands

Although RealFlash is best suited for linear presentations, you can add interactivity through the Shockwave commands listed below. RealFlash maps these commands to RealPlayer functions. At the end of your RealFlash clip, for example, you might have a graphic that says, "Click here to visit our home page." The Shockwave `Get URL` command used with this graphic corresponds to an internal RealPlayer command that displays the URL in the browser window.

Shockwave and RealPlayer Interactive Commands

Shockwave Command	RealPlayer Mapping	Action
`Play`	`Play`	Playback begins or resumes.
`Stop`	`Pause`	Clip pauses until action is performed or the Play button is pressed.
`Goto`	`Seek and Pause`	RealPlayer seeks to the designated frame, buffers the clip preroll, and pauses.
`Goto and Play`	`Seek and Play`	RealPlayer seeks to the designated frame, buffers the clip preroll, and begins playback.
`Get URL`	(internal)	Displays URL in browser window. Because the user has to return to RealFlash manually, use this only at the end of a clip.

Tip

Because seeking requires buffering, do not use `Goto` commands to advance from one scene to the next. When you export your animation to Shockwave Flash, scenes are concatenated so that the animation flows from one scene to the next.

Additional Information

A SMIL file can define clickable hyperlinks for the presentation. See "Linking to Other Media" on page 104.

Creating a RealFlash Clip

The following sections explain the steps for creating a RealFlash clip once you finish developing your animation in Macromedia Flash. Refer to the manuals for the tools you use for step-by-step instructions on carrying out each task.

Exporting Shockwave Flash

You export your animation to a Shockwave Flash 2.0 file (`.swf`) for use with RealSystem. This creates a compressed version of the animation suitable for streaming. When you export the Shockwave Flash file, you disable the audio stream. You later export the soundtrack separately and convert it to RealAudio. Here are tips on exporting Shockwave Flash:

- Set JPEG quality between 30 and 50. This helps to keep the file size down.
- Click the **Generate Size Report** checkbox. This creates a movie report you can use with the tuning spreadsheet. RealNetworks recommends that you use the movie report and spreadsheet to evaluate your RealFlash clip for bandwidth efficiency.

The ratio of Shockwave file size to clip length is a good indication of the overall bandwidth requirement. Convert the file size to Kilobits and divide by the number of seconds in the animation to get the average bandwidth. This number should be below your allowable bit rate for Shockwave Flash. For example, to find the average bandwidth of a 325 Kilobyte file that plays for 3

minutes, multiply 325 by 8 to get 2624 Kilobits. Then divide by 180 seconds to get an average bandwidth of 14.6 Kbps.

Converting File Size to Kilobits

Using This Measurement	Do This to Get Kilobits
Megabytes	Multiply by 8192
Kilobytes	Multiply by 8
bytes	Divide by 128
bits	Divide by 1024

Tip

You can also find the average bit rate of a Shockwave Flash file by dropping it onto RealPlayer G2 and observing the RealPlayer status bar.

Keep in mind that even a clip with an acceptable average bandwidth may stall during playback because it contains bandwidth spikes. The spreadsheet indicates where spikes occur.

Tuning Shockwave Flash

After you generate a Shockwave Flash file and create the movie report, you must use the RealFlash Bandwidth Tuner to view bandwidth statistics and adjust the file's streaming bit rate. You can also use the RealFlash Bit Rate Calculation Spreadsheet to examine the file's bandwidth consumption frame-by-frame. You will need Microsoft Word and Microsoft Excel to use the spreadsheet. See the tuner online help for instructions on using the tuner and spreadsheet.

Note

These tools are included in the RealSystem G2 Authoring Kit, available through registration at **http://www.real.com/products/tools/authkit/index.html**.

Exporting Audio

After you have created and tuned your Shockwave Flash file, export the movie soundtrack as a Windows AVI or Macintosh QuickTime file, setting 32x21 as the height and width attributes to minimize disk space use. Next, encode the

AVI or QuickTime soundtrack in the RealAudio format with a RealNetworks encoding tool, using the file extension .rm.

Additional Information

For more on audio production, see Chapter 4 beginning on page 39. For codec considerations in RealFlash, see "Dividing Bandwidth Between Shockwave Flash and RealAudio" on page 65.

Delivering the RealFlash Presentation

When your Shockwave Flash and RealAudio files are complete, you create a SMIL file that lists the URLs for these files. Chapter 7 beginning on page 75 explains how to create the SMIL file. In its simplest form, the SMIL file specifies that the two files play in parallel:

```
<smil>
  <body>
    <par>
      <audio src="rtsp://realserver.company.com:554/media/sound.rm"/>
      <animation src="rtsp://realserver.company.com:554/media/cartoon.swf"/>
    </par>
  </body>
</smil>
```

You can also use the SMIL file to define clickable hypertext links, create timing offsets between the clips, or add presentation information such as title, author, and copyright. Next, move the Shockwave Flash, RealAudio, and SMIL files to RealServer and link your Web page to the SMIL file.

Additional Information

For instructions on linking Web pages to clips on RealServer, see "Streaming Clips from RealServer G2" on page 146.

Supporting RealPlayer 5.0

As described in "Adding RealPlayer 5.0 Compatibility" on page 66, you can support RealPlayer 5.0 in a number of ways. The simplest means of delivering a 5.0-compatible presentation through a single URL is to create a SMIL file for RealPlayer G2 as described above. Then create a Ram file (extension .ram) that looks like this:

```
rtsp://realserver.company.com:554/media/animation.smil
--stop--
pnm://realserver.company.com:7070/media/cartoon.swf+sound.rm
```

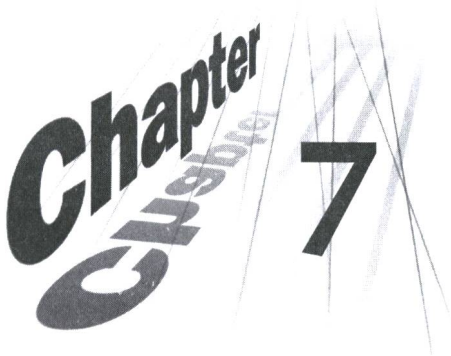
You next link your Web page to this Ram file with a standard HTTP hypertext link. Users with RealServer G2 receive the SMIL-based presentation. Those with RealPlayer 5.0 receive the presentation defined below the `--stop--` marker. Note that this second link uses the older PNA protocol and associates the animation and soundtrack through the "+" operator.

Additional Information

See "Creating a Ram File Manually" on page 153. To embed the presentation in a Web page, see Chapter 8 beginning on page 117.

Note

In the examples above, the same soundtrack (`sound.rm`) is used for both RealPlayer 5.0 and RealPlayer G2. This works if the soundtrack uses a RealAudio 5.0 codec or is a RealAudio G2 SureStream clip with the 5.0 backwards-compatibility option.



ASSEMBLING A PRESENTATION WITH SMIL

When your multimedia presentation contains multiple clips—such as a slideshow and a video played together—you use Synchronized Multimedia Integration Language (SMIL) to coordinate the parts. Pronounced "smile," SMIL uses a simple but powerful mark-up language for specifying how and when clips play. After writing the SMIL file, you put it on RealServer and link your Web page to it as described in Chapter 10, beginning on page 145.

Tip

If you have just one clip in your presentation, such as a single RealVideo clip, you don't need to create a SMIL file. Just link your Web page to the media clip as explained in Chapter 10.

Additional Information

Once you are familiar with SMIL, refer to "Appendix C: SMIL Quick Reference" beginning on page 181.

Creating a SMIL File

You can create a SMIL file (extension .smil) with any text editor or word processor that can save output as plain text. If you are familiar with HTML mark-up, you will pick up SMIL quickly. In its simplest form, a SMIL file lists multiple media clips played in sequence:

```
<smil>
  <body>
    <audio src="rtsp://realserver.company.com/one.rm"/>
    <audio src="rtsp://realserver.company.com/two.rm"/>
    <audio src="rtsp://realserver.company.com/three.rm"/>
  </body>
</smil>
```

SMIL General Rules

SMIL has many similarities to HTML, but also some important differences. When you create your SMIL file, keep the following general rules in mind.

Mark-up Starts with <smil> and Ends with </smil>

The SMIL mark-up must start with a <smil> tag and end with the </smil> closing tag. All other mark-up appears between these two tags:

```
<smil>
 ...all other SMIL mark-up...
</smil>
```

Body Section is Required but Header Section is Optional

A SMIL file can include an optional header section defined by <head> and </head> tags. It requires a body section defined by <body> and </body> tags:

```
<smil>
  <head>
    ...optional section with all header mark-up...
  </head>
  <body>
    ...required section with all body mark-up...
  </body>
</smil>
```

The header section is used to specify presentation information, as noted in "Adding Presentation Information" on page 91, and to define clip layout, as described in "Laying Out Multiple Clips" on page 95.

Lowercase Text for Tags and Attributes

SMIL tags and attributes must be lowercase.

Some Tags Must Close with a Forward Slash

A tag that does not have a corresponding end tag (for example, the <smil> tag has the end tag </smil>), must close with a forward slash. For example:

```
<audio src="first.rm"/>
```

Attribute Values Enclosed in Double Quotation Marks

Attribute values, such as "first.rm" shown above, must be enclosed in double quotation marks. File names in SMIL must reflect the file name exactly as it appears on the server. They can use upper, lower, or mixed case.

SMIL Files Use the Extension .smi or .smil

Save your SMIL file with the extension .smi or .smil. The latter extension (.smil) is preferred to avoid possible conflict with other files. Do not include spaces in the file name. For example, you can have the file my_presentation.smil but not the file my presentation.smil.

Coded Characters Used in Header

You need to use codes to add quotation marks, apostrophes, ampersands, or angle brackets to content in a SMIL header, such as a presentation title. See "Using Coded Characters" on page 94.

HTML-Style Comments Allowed

As in HTML, the SMIL comment tag starts with **<!--** and ends with **-->**. The ending does not include a forward slash:

```
<!-- This is a comment -->
```

Indentation Recommended

Although examples in this chapter indent SMIL tags to various levels to illustrate the SMIL structure, indentation is not required. Indenting your SMIL files like the examples shown below will help you keep track of the SMIL functions, though.

Specifying Clip Locations

To add a clip to the presentation, you include in the SMIL body section a clip source tag that describes the clip type and location:

```
<audio src="rtsp://realserver.company.com/audio/first.rm"/>
```

The tag begins with one of the clip type attributes listed in the following table.

Source Clip Attributes

Clip Attribute	Used For
animation	Animation clips, such as Shockwave Flash (.swf) used in RealFlash.
audio	Audio clips such as RealAudio (.rm).
img	JPEG (.jpg) or GIF images (.gif). See also "Defining Image Options" on page 111.
ref	Any clip type not covered by other attributes, such as a RealPix file (.rp).
text	Static text clips (.txt).

(Table Page 1 of 2)

Source Clip Attributes (continued)

Clip Attribute	Used For
textstream	Streaming RealText clips (.rt).
video	Video or other clips that display continuous motion, such as RealVideo (.rm).

(Table Page 2 of 2)

Additional Information

For a list of supported streaming formats, see "Choosing Media to Stream" on page 11.

Although a source clip tag must start with a clip type attribute, attributes do not affect playback because RealPlayer determines the clip type through other means. Specifying a text clip as audio, for example, does not adversely affect playback. Although using the different clip attributes helps you keep track of the media types, you could specify all clips with ref, for example.

After the clip type attribute, the src attribute lists the clip location. How you specify this location depends on whether you will stream the presentation with RealServer, download the clip from a Web server, or play clips back from a local computer.

Linking to Clips on RealServer

When a RealSystem G2 presentation streams over a network, the media clips reside on RealServer. Each source clip's src attribute gives the clip's URL:

```
<audio src="rtsp://realserver.company.com:554/audio/first.rm"/>
```

The following table explains the URL components in the example above. Contact your RealServer administrator to get the RealServer address, RTSP port, and directory structure.

RealServer URL Components

Component	Meaning
rtsp://	RTSP streaming protocol. In contrast, URLs in Web pages start with http://. Although RealServer also supports HTTP, RealServer clips should always use RTSP.
realserver.company.com	RealServer address. This varies for each RealServer. It typically uses an identifier such as realserver instead of www. Or it may use a TCP/IP address such as 172.2.16.230 instead of a name.

(Table Page 1 of 2)

RealServer URL Components (continued)

Component	Meaning
:554	RealServer port for RTSP connections. Port 554 is the default, so you can leave this out of URLs unless the RealServer administrator changed the RTSP port. If the port number is required, separate it from the address with a colon.
/audio/	RealServer directory that holds the clip. The directory structure may be several levels deep.
first.rm	Clip file name.

(Table Page 2 of 2)

Creating Relative URLs

If your presentation includes many clips that are on the same server, you can make each URL relative to a base target that you define in the header:

```
<head>
  <meta name="base" content="rtsp://realserver.company.com/"/>
</head>
<body>
  <audio src="audio/first.rm"/>
  <audio src="audio/second.rm"/>
  <audio src="rtsp://realserver.real.com/audio/third.rm"/>
</body>
```

Because the third clip has a full URL specified for it, the base target is ignored. For the first two clips, however, the src values are appended to the base target, effectively giving the clips these URLs:

```
rtsp://realserver.company.com/audio/first.rm
rtsp://realserver.company.com/audio/second.rm
```

If no base target is given, RealPlayer assumes that the clip paths are relative to the location of the SMIL file. In the example above, therefore, you could leave the base target out if the SMIL file itself resides in a directory that contains the audio subdirectory that in turn holds the RealAudio clips.

Tip

The relative syntax for SMIL files works like relative links in HTML, so you can use directory notation such as "../". You can find additional information about this topic in an HTML reference guide.

Linking to Clips on a Web Server

To use a clip hosted on a Web server, use a standard HTTP URL such as the following in a clip source tag:

```
<img src="http://www.company.com/images/logo.gif"/>
```

URLs to clips on a Web server can also be relative to a base target as described above. Keep in mind that although a Web server can host any clip, a Web server cannot perform all the functions of RealServer.

Additional Information

See "Hosting a Presentation on a Server" on page 21 and "Limitations on Web Server Playback" on page 151.

Linking to Local Clips

If your presentation clips will reside on the user's local computer (as with a multimedia tutorial included with a software application, for example), you include the SMIL file locally as well. The src attributes in the SMIL file list presentation clips in this format:

```
src="audio/first.rm"
```

This example is a local, relative link to a clip that resides one level below the SMIL file in the audio directory. For local access, you typically want to use relative links because you cannot be sure where users will place clips on their machines.

Alternately, you can use absolute, local links to specify exact locations. The syntax for absolute links is the same as with HTML. It varies with operating systems, however, and you should be familiar with the directory syntax for the system you're using. For example, the following absolute link syntax works for Windows machines, but not UNIX or Macintosh:

```
src="file://c:\audio\first.rm"
```

Warning

Microsoft Internet Explorer 3.0 tries to display local SMIL files as HTML. To support this browser, omit the <head> tag or launch the presentation through a Ram file as described in "Creating a Ram File Manually" on page 153. This problem does not occur with Netscape Navigator or Internet Explorer 4.0. Nor does it occur

when you stream clips from a server to Internet Explorer 3.0.

Grouping Clips

With the SMIL <seq> and <par> tags, you can create groups that structure your presentation. The following sections explain how to play clips in sequence or parallel, as well as how to repeat groups or clips within groups.

Playing Clips in Sequence

Use the <seq> tag to play clips in sequence. In the following example, the second clip begins when the first clip finishes.

```
<seq>
  <audio src="audio/newsong.rm"/>
  <audio src="audio/oldsong.rm"/>
</seq>
```

If your presentation included just the clips above, you wouldn't need to use the <seq> tag. You could simply list the clips in order and RealPlayer would play them in sequence. The <seq> tag is most commonly combined with <par> to create combinations of sequential and parallel clips.

Playing Clips in Parallel

You can play two or more clips at the same time through the <par> tag. For example, the following combines a RealVideo clip with a RealText clip:

```
<par>
  <video src="videos/newsong.rm"/>
  <textstream src="lyrics/newsong.rt"/>
</par>
```

When you play clips in parallel, be sure that they do not exceed the presentation bandwidth as described in "Choosing a Target Bandwidth" on page 29. When RealServer G2 streams parallel groups, it ensures that the clips stay synchronized. If some video frames don't arrive, for example, RealServer either drops those frames or halts playback until the frames arrive.

Additional Information

When multiple visual clips play in parallel, you need to define each clip's playback region. For more

information, see "Laying Out Multiple Clips" on page 95.

Ending a Parallel Group on a Specific Clip

By default, a <par> group ends once all clips finish playing back. You can modify this with an end timing value as described in "Setting Begin and End Times" on page 85. Or you can use the endsync attribute to stop the group when a specific clip finishes playback:

```
<par endsync="first">
  ...
</par>
```

first

Use the attribute endsync="first" to stop the <par> group the first time a clip in the group ends playback. All other clips in the group stop playing at that point regardless of their playback status and any timing parameters specified for them.

last

The attribute endsync="last" causes the <par> group to conclude when all clips have finished playing. Because this is the default value, you can omit the endsync attribute from the <par> tag to achieve this effect.

id(clip id)

The attribute endsync="id(*clip id*)" causes the <par> group to conclude when the specified clip reaches its end. All other clips in the group stop playing at that point regardless of their playback status and any timing parameters specified for them. The specified clip must have a corresponding id value in its source tag:

```
<par endsync="id(vid)">
  <video id="vid" src="videos/newsong.rm"/>
  <textstream src="lyrics/newsong.rt"/>
</par>
```

Repeating a Clip or Group

The repeat="n" attribute makes a clip or group play a specified number of times. You can add it to a clip source tag, for example:

```
<video src="videos/newsong.rm" repeat="4"/>
```

You can also add it to a <seq> or <par> group to make the entire group repeat:

```
<par repeat="3">
  <video src="videos/newsong.rm"/>
  <textstream src="lyrics/newsong.rt"/>
</par>
```

In the following example, a video clip within a <par> group plays twice. Assuming the video lasts longer than the text stream, the <par> group ends when the video concludes its second playback:

```
<par>
  <video src="videos/newsong.rm" repeat="2"/>
  <textstream src="lyrics/newsong.rt"/>
</par>
```

In the following <seq> group, the second video plays only after the first video plays twice:

```
<seq>
  <video src="videos/newsong.rm" repeat="2"/>
  <video src="videos/newsong2.rm"/>
</seq>
```

Combining <seq> and <par> Tags

You can combine and nest <seq> and <par> tags as needed. Note that the organization of these tags greatly affects the presentation playback:

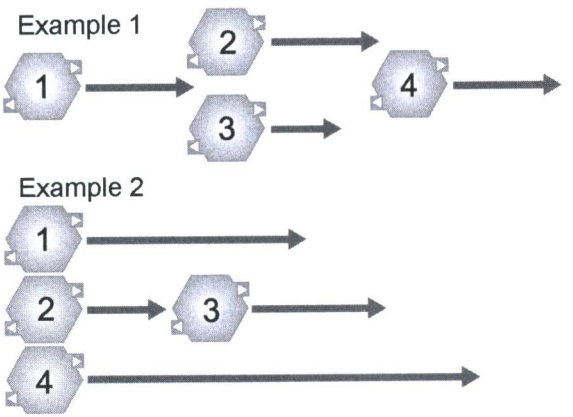
```
<seq>
   clip 1
   <par>
     clip 2
     clip 3
   </par>
   clip 4
</seq>
```

In the example above, clip 1 plays first. When it finishes, clip 2 and clip 3 play together. When both clip 2 and clip 3 have finished, clip 4 plays. You get very different results, though, if you switch the <seq> and <par> groupings:

```
<par>
   clip 1
   <seq>
     clip 2
     clip 3
   </seq>
   clip 4
</par>
```

In this example, clip 1, clip 2, and clip 4 all begin at the same time. When clip 2 finishes, clip 3 starts. The following figure illustrates the difference between these different groupings.

Different Playback Results with <seq> and <par> Groups



Specifying Timing

SMIL timing elements let you specify when a clip or group starts playing and how long it plays. All timing elements are optional. If you do not set them, clips start and stop according to their normal timelines and their positions within <par> and <seq> groups. The easiest way to designate a time is with shorthand markers of h, min, s, and ms as illustrated in the following table.

Timing Shorthand Examples

Shorthand Example	Value
2.5h	2 hours, 30 minutes
2.75min	2 minutes, 45 seconds

(Table Page 1 of 2)

Timing Shorthand Examples (continued)

Shorthand Example	Value
15.55s	15 seconds, 550 milliseconds
670.2ms	670.2 milliseconds

(Table Page 2 of 2)

Tip

Decimal values are not required. You can express two seconds as "2s" or "2.0s", for example.

Normal Play Time Format

You can also express time elements in a hh:mm:ss.xy format. Here, hh is hours, mm is minutes, ss is seconds, x is tenths of seconds, and y is hundredths of seconds. In this example:

```
begin="02:34.0"
```

the time value is 2 minutes, 34 seconds. If the value does not include a decimal point, RealPlayer takes the last value to be seconds. So it reads the following value as 2 minutes, 34 seconds rather than as 2 hours, 34 minutes:

```
begin="02:34"
```

Setting Begin and End Times

The begin attribute works for any clip or group. You can use it to start a clip at a specific point within the timeline:

```
<video src="videos/newsong.rm" begin="20.5s"/>
```

Were this clip in a <par> group, the begin attribute would start the clip playing back 20.5 seconds after the group becomes active. If the clip were in a <seq> group, the attribute would delay the clip's normal playback by inserting 20.5 seconds of blank time before the clip starts. Hence timing is relative to the start of the <seq> or <par> group, not the start of the overall presentation.

Additionally, you can set an end attribute alone or combined with a begin attribute as shown here:

```
<video src="videos/newsong.rm" begin="20.5s" end="62.7s"/>
```

In this example, the clip ends at 62.7 seconds into its part of the presentation timeline, playing a total of 42.2 seconds regardless of the length of its internal timeline. If the video's internal timeline is shorter (30 seconds, for example) the fill attribute determines what happens onscreen after the video stops playing but before the end time is reached.

Additional Information

"Setting a Fill" on page 88.

Using Begin and End Times with Groups

You can use begin or end with a <par> or <seq> group:

```
<par begin= "5s" end= "3.5min">
  ...
</par>
```

This begin value delays group playback until 5 seconds after the preceding group or clip finishes. The end time means all clips in the group stop playing after 3.5 minutes regardless of their states. If all clips reach their normal conclusion by 3 minutes and 20 seconds after the group starts, for example, the next group or clip starts after 10 seconds of blank time.

If you use an end time and a repeat attribute as described in "Repeating a Clip or Group" on page 82, the group repeats only after the end time elapses. You should not set an end time along with an endsync attribute in a <par> group, as this sets up conflicting end times.

Additional Information

For more on endsync, see "Ending a Parallel Group on a Specific Clip" on page 82.

Setting Internal Clip Begin and End Times

The clip-begin and clip-end attributes specify a clip's internal timing marks where playback begins and ends. You can use them with clips that have internal timelines, such as audio, video, and animation. Do not use them with groups or static clips such as still images. Here is an example:

```
<video src="videos/newsong.rm" clip-begin="10.5s" clip-end="50.7s"/>
```

Here, the clip starts playing at its internal 10.5-second mark rather than at its normal beginning. It stops when it reaches its 50.7-second mark, playing for a total of 40.2 seconds.

Note

Do not use clip-begin and clip-end for a live broadcast or when playing clips back from a Web server. For more information, see "Limitations on Web Server Playback" on page 151 and "Using SMIL with a Broadcast" on page 164.

Combining clip-begin and clip-end with begin and end

You can combine clip-begin and clip-end attributes with begin and end attributes. Here, a begin time has been added to the clip example shown above:

```
<video src="videos/newsong.rm" clip-begin="10.5s" clip-end="50.7s" begin="5s"/>
```

The begin time delays the clip's normal starting point by five seconds. When this time elapses, the clip starts at its 10.5 second internal timeline marker, then plays for 40.2 seconds. In this case, the clip-end attribute determines how long the video is active. But you could also add an end attribute as shown here to modify this behavior:

```
<video src="videos/newsong.rm" clip-begin="10.5s" clip-end="50.7s" begin="5s"
  end="50s"/>
```

Combined with begin, the end value of 50 means the clip's "window" within the presentation is 45 seconds. Because the clip stops playing after 40.2 seconds, there is an extra 4.8 seconds during which the clip does not play but remains active. How the video window appears during these final seconds depends on the fill attribute.

Additional Information

"Setting a Fill" on page 88.

Setting Durations

The dur attribute controls how long a clip or group appears after it starts to play back. It is useful with graphic images, as shown in this example:

```
<img src="graphics/poster.jpg" dur="14.5s"/>
```

This dur time makes the graphic disappears 14.5 seconds after it appears. You can also use dur in place of end. For example, in the following clip:

```
<video src="videos/newsong.rm" begin="20.5s" end="62.7s"/>
```

you can substitute a dur attribute for the end attribute to achieve the same result:

```
<video src="videos/newsong.rm" begin="20.5s" dur="42.2s"/>
```

In both examples, the clips stop playback 42.2 seconds after starting. With the end attribute, the total playing time is the end value minus the begin value. The dur attribute ignores the begin value, stopping the clip 42.2 seconds after it starts. Use either end or dur, therefore, depending on how you want to measure time. Do not use both attributes in the same tag, however.

Additional Information

The dur attribute can function like end in a group, too. For more information, see "Using Begin and End Times with Groups" on page 86.

Setting a Fill

The fill attribute determines what happens to the clip immediately after it plays to its normal end point, or its specified end time or duration elapses. The fill value can be remove or freeze.

remove

The default value fill="remove" removes the clip. When this attribute is used with a still image, the image disappears once the end time has elapsed.

freeze

Use fill="freeze" to freeze the clip on its last frame. When used with a video, the video's last frame stays on the screen. Suppose that you have a 20-second video and specify a 30-second duration with freeze:

```
<video src="videos/newsong.rm" dur="30s" fill="freeze"/>
```

After the video plays, its last frame displays for 10 seconds. The video disappears when the end time elapses. The fill="freeze" attribute has no effect on audio. Do not use fill="freeze" for a graphic image that also uses a dur attribute.

Clip Timing Example

The following example shows two audio clips with different timing options:

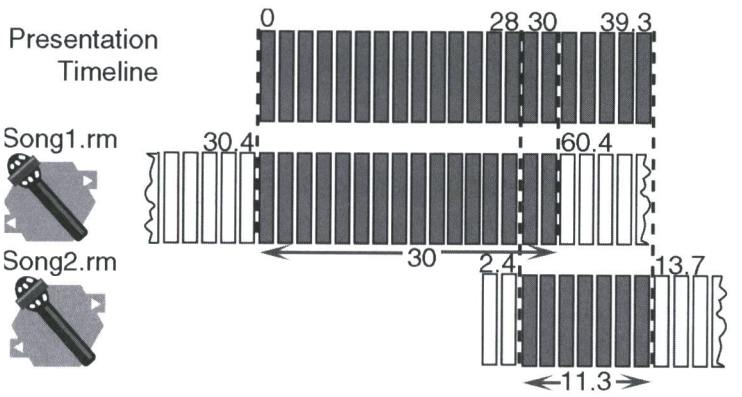
```
<par>
  <audio src="song1.rm" clip-begin="30.4s" dur="30s"/>
  <audio src="song2.rm" begin="28s" clip-begin="2.4s" clip-end="13.7s"/>
</par>
```

The timing options modify the <par> tag so that the two clips start at different times. The first clip begins to play immediately, but starts at 30.4 seconds into its timeline, playing for exactly 30 seconds.

The second clip is delayed for 28 seconds. That means it overlaps the first clip by 2 seconds. It starts at 2.4 seconds into its timeline and ends at 13.7 seconds into its timeline, thus playing for 11.3 seconds. The total playing time for this group is 30 seconds for the first clip, plus 11.3 seconds for the second clip,

minus the 2 second overlap: 39.3 seconds. The following figure illustrates the relationships of the clip timelines to the overall presentation timeline.

Clip Timing Example



Switching Between Alternate Choices

With the <switch> tag, you can specify multiple options that RealPlayer can choose between. The <switch> group specifies any number of choices in this form:

```
<switch>
  <choice1 test-attribute="value1"/>
  <choice2 test-attribute="value2"/>
  ...
</switch>
```

RealPlayer looks at choices in order, evaluating each test attribute and its value to determine which clip to choose. The choices are typically clip source tags such as <video *test-attribute="value"*/>, but RealPlayer can also choose between groups when test attributes appear in <par> or <seq> tags.

Note

Every RealPlayer must have a viable choice within a <switch> statement. RealPlayer will not play any clip in the <switch> group if it finds no satisfactory test attribute. The following sections explain how to guarantee that every RealPlayer has a viable option.

Setting Language Choices

When the <switch> group test attribute is system-language, each source clip is for a different language. The following example shows a video slideshow with separate audio narrations in French, German, Spanish, and English. Based on the language preference set through its **Options** menu, as well as the system-language code defined in the SMIL file, RealPlayer chooses a clip to play:

```
<par>
 <video src="slides/seattle.rm"/>
 <!-- select audio based on RealPlayer language preference setting -->
 <switch>
  <audio src="french/seattle.rm" system-language="fr"/>
  <audio src="german/seattle.rm" system-language="de"/>
  <audio src="spanish/seattle.rm" system-language="es"/>
  <audio src="english/seattle.rm"/>
 </switch>
</par>
```

Because the last option does not have a test attribute, a RealPlayer that does not have French, German, or Spanish explicitly set as its preferred language chooses the English clip regardless of its actual language setting. RealPlayer evaluates options in order, so the last option should be, as shown in this example, a default language that applies if no other option is viable. If you list no default option and certain RealPlayers prefer languages other than the ones you list, those RealPlayers will not play any of the clips.

Additional Information

Appendix E starting on page 195 lists the system-language codes such as "fr" you use to designate content in different languages.

Setting Bandwidth Choices

To serve different clips to viewers with different connection speeds, use the <switch> tag to define options each RealPlayer can choose based on its available bandwidth. As shown below, you can group clips with <par> tags, using the system-bitrate attribute to list the approximate bandwidth (in Kbps) each group consumes:

```
<switch>
 <par system-bitrate="75000">
  <!--for dual isdn and faster-->
  <audio src="audio/newsong1.rm"/>
```

```
    <video src="video/newsong1.rm"/>
    <textstream src="lyrics/newsong1.rt"/>
  </par>
  <par system-bitrate="47000">
    <!--for single isdn-->
    <audio src="audio/newsong2.rm"/>
    <video src="video/newsong2.rm"/>
    <textstream src="lyrics/newsong2.rt"/>
  </par>
  <par system-bitrate="20000">
    <!--for 28.8 modems-->
    <audio src="audio/newsong3.rm"/>
    <video src="video/newsong3.rm"/>
    <textstream src="lyrics/newsong3.rt"/>
  </par>
</switch>
```

Always list system bandwidth options from highest to lowest. RealPlayer evaluates options in the order listed, selecting the first viable option even if subsequent options suit it better. So if the 28.8 Kbps option is first, a RealPlayer with a dual-ISDN connection will choose that option because it is the first viable option listed.

Also ensure that the last option satisfies the lowest bandwidth connection you want to support. If you do not list an option suitable for 28.8 Kbps modems, for example, RealPlayers connected through those modems will not play the presentation.

Additional Information

"Writing Complex SMIL Switch Statements" on page 176 explains how to use <switch> with SureStream clips, as well as how to test for both language and bandwidth.

Adding Presentation Information

The SMIL file header can use <meta> tags to list presentation information such as title, author, and copyright:

```
<head>
  <meta name="title" content="Bob and Susan Discuss Streaming Media"/>
  <meta name="author" content="RealNetworks Media Productions"/>
  <meta name="copyright" content="(c)1998 RealNetworks"/>
  <meta name="abstract" content="Bob and Susan, two Internet technology
experts, discuss the future of streaming media."/>
 </head>
```

This example defines a title, author, copyright, and abstract. You can define other values as well, such as an e-mail address, simply by adding an attribute such as `name="email"`. This information displays when the user gives the RealPlayer **Help>About this Presentation** command. In addition, the title displays at the top of the RealPlayer window and in the run list under the RealPlayer **File** menu.

Note

Name values, as in `name="title"`, must be lowercase. When defining long content such as an abstract, don't use line breaks or tabs within a `content` value.

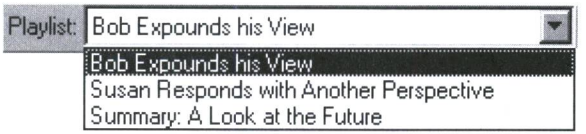
Managing the Playlist

In addition to the presentation information, RealPlayer keeps a playlist of clip titles. Through this playlist, users can play different parts of the SMIL presentation. The following example shows the same header information as above, but adds titles for each video clip in the body section:

```
<smil>
  <head>
    <meta name="title" content="Bob and Susan Discuss Streaming Media"/>
    <meta name="author" content="RealNetworks Media Productions"/>
    <meta name="copyright" content="(c)1998 RealNetworks"/>
    <meta name="abstract" content="Bob and Susan, two Internet technology
experts, discuss the future of streaming media."/>
  </head>
  <body>
    <seq>
      <video src="clip1.rm" title="Bob Expounds his View"/>
      <video src="clip2.rm" title="Susan Responds with Another Perspective"/>
      <video src="clip3.rm" title="Summary: A Look at the Future"/>
    </seq>
  </body>
</smil>
```

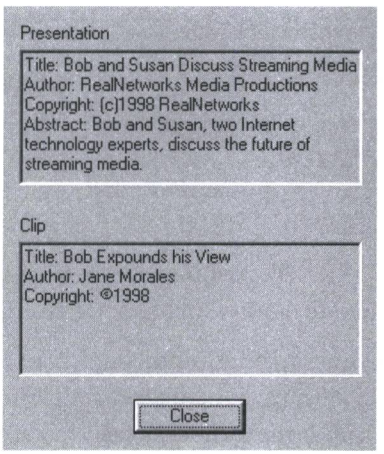
The titles for the individual clips appear in RealPlayer's pull-down playlist as illustrated in the following figure. The user can also view the playlist with RealPlayer's **File>Playlist** command.

RealPlayer's Pull-Down Playlist Showing Clip Titles



Only the clip titles affect the playlist, but you can also define an author, copyright, and abstract for each clip. As illustrated below, clip information appears when the RealPlayer user gives the **Help>About this Presentation** command while the clip plays.

Presentation and Clip Information



Note that in this figure, the clip information section shows the clip title defined in the SMIL file, but also includes author and copyright information not defined through SMIL. The author and copyright information is encoded in the clip.

Tips for Defining Clip Information

The following points explain the relationship between clip information set through SMIL and information encoded in a clip.

- Clips may include their own title, author, and copyright information. With RealVideo, for example, this information is encoded within the clip. For RealPix, it is set within the RealPix text file.

- You can override any title, author, or copyright information encoded in a clip by adding title, author, and copyright attributes to the clip's SMIL source tag:

  <ref src="..." title="*title*" author="*name*" copyright="*copyright*"/>

- To make the playlist useful, always define a title for each clip through SMIL. RealPlayer reads the SMIL titles when the presentation begins. It reads a clip's encoded title only when the clip starts playback. So if you define titles through SMIL, RealPlayer can immediately build a playlist that allows the user to choose different clips.

- You can set a clip abstract only through SMIL by adding the abstract attribute and its appropriate value to the clip's source tag:

  <ref src="..." abstract="*abstract*"/>

- Only title, author, copyright, and abstract attributes are supported for clips. For the entire presentation, however, you can define any information through the header section <meta.../> tags. See "Adding Presentation Information" on page 91 for more information.

- For a parallel group, use title, author, copyright, and abstract attributes in the <par> tag instead of the clip source tags. Clip titles are ignored, and only the group title shows in the playlist. For more on <par>, see "Playing Clips in Parallel" on page 81.

- A title, author, copyright, or abstract attribute in a <seq> tag is ignored. This information must be defined for individual clips or parallel groups.

- Although it's easier to set clip information through SMIL, it's good practice to encode the title, author, and copyright information in the clip as well. This preserves the clip information in case you stream the clip without using a SMIL file.

Using Coded Characters

In a header, SMIL interprets quotation marks, apostrophes, ampersands, and angle brackets as syntax markers.To have these characters show up as text in RealPlayer, you use codes in the header. As shown in the following table, codes

begin with an ampersand ("&") and end with a semicolon (";"). SMIL interprets these codes the same way as popular Web browsers.

SMIL Coded Characters

Code	Character	Example
"	quotation mark	"
&	ampersand	&
'	apostrophe	'
<	left angle bracket ("less than" sign)	<
>	right angle bracket ("greater than" sign)	>

For example, to add the following as a title:

```
"Multimedia's <smil> & you"
```

You enter this in the SMIL file header:

```
<meta name="title" content=
 "&quot;Multimedia&apos;s &lt;smil&gt; &amp; you&quot;"/>
```

Laying Out Multiple Clips

If your presentation plays only one clip at a time, you do not need to create a layout. Each clip automatically plays in the main RealPlayer window, the window resizing automatically for each new clip. If you want to keep the playback area stable in size as different clips play back, or if your presentation displays several clips at a time, you can define playback areas called "regions" within the main RealPlayer window:

1. In the SMIL file header, you create a `<layout>` group, using `<region>` tags to name playback regions and define their sizes and locations within the RealPlayer main window. See "Defining the Layout" below.
2. In the SMIL file body, you use `region` attributes to specify which source clips play in which regions. See "Assigning Clips to Regions" on page 103.

Additional Information

See "Laying out SMIL Presentations" on page 132 for instructions on using RealPlayer's Netscape plug-in or ActiveX control to lay out the presentation in a Web page instead of in RealPlayer.

Defining the Layout

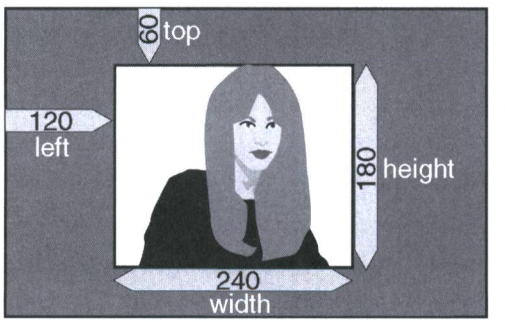
When you lay out regions, you create a root-layout region that defines the size of the RealPlayer main window. You then create other regions using a simple coordinate system measured across and down from the top, left-hand corner of the root-layout region. All measurements are in pixels or percentages, with zero pixels as the default. The following table lists the attributes that define region size and placement.

Region Size and Placement Attributes

Attribute	Sets	Pixel Example	Percentage Example
top	offset from top of window	top="60"	top="10%"
left	offset from left side of window	left="120"	left="20%"
width	region width	width="240"	width="40%"
height	region height	height="180"	height="30%"

The next figure illustrates the layout of a single playback region within a root-layout region.

Basic Region Layout



Setting the Root Layout Region

With the <root-layout.../> tag, you specify the size of the entire playback area in pixels (percentages are not accepted for the root-layout region). You cannot display images or play clips in the root-layout region, as it is meant to set the overall playback area. The example shown below creates a root-layout region 250 pixels wide by 230 pixels high. When the presentation begins, the RealPlayer window expands to this size. Other regions measure their top and left offsets from the upper, left-hand corner of this root-layout region:

```
<head>
  <layout>
    <root-layout width="250" height="230"/>
    ...other regions defined here...
  </layout>
</head>
```

Note

Although you can omit root-layout to have RealPlayer calculate the playback area based on the sizes of the other regions, it is better to define root-layout to avoid unexpected results.

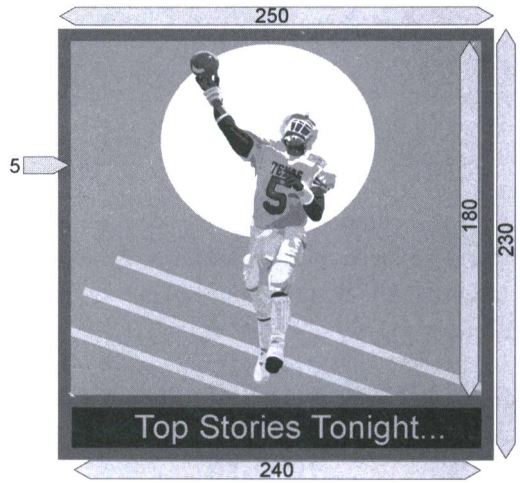
Defining Playback Regions

You create playback regions for media clips with <region> tags. These regions must lay within the root-layout region. Any part of a region that lays outside the root-layout region is cut off. Each <region.../> tag must define top, left, width, and height attributes. The example below defines two regions named "videoregion" and "textregion":

```
<head>
  <layout>
    <root-layout background-color="maroon" width="250" height="230"/>
    <region id="videoregion" top="5" left="5" width="240" height="180"/>
    <region id="textregion" top="200" left="5" width="240" height="20"/>
  </layout>
</head>
```

In this example, both regions are offset 5 pixels to the right of the root-layout region's left edge. The video region displays 5 pixels down from the top of the root-layout region, and the text region displays 200 pixels down. The following figure illustrates this placement.

SMIL Sample Layout with Video and Text Windows



Tips for Defining Regions

Note the following about SMIL regions:

- All regions used in the presentation must be defined in the header. All regions are created at the beginning of the presentation, so regions containing background colors as described in "Adding a Background Color" on page 99 may obscure other regions they overlap.
- Using region transparency and the z-index attribute, you can emulate dynamic creation and destruction of regions. For details, see "Hiding Regions with z-index" on page 174.
- To use a graphic image such as a GIF or a JPEG as a background, define a playback region the same size as the root-layout region. Display the image in this region and use the z-index attribute to make it display behind other regions.

Additional Information

See "Ordering Overlapping Regions with z-index" on page 101.

Using Percentage Values for Regions

For a region's height, width, and offset measurements, you can use percentages that reflect a fraction of the root-layout region's size. The following example uses percentages to define playback areas similar to those shown in the sample above:

```
<head>
  <layout>
    <root-layout background-color="maroon" width="250" height="230"/>
    <region id="videoregion" top="2%" left="2%" width="96%" height="78%"/>
    <region id="textregion" top="80%" left="2%" width="96%" height="18%"/>
  </layout>
</head>
```

Tips for Using Percentage Values

Note the following when using percentage values to define regions:

- You must include a root-layout region defined in pixels when specifying region measurements in percentages.
- You can mix pixels and percentages. You could define the `top` and `left` offset measurements in percentages, for example, while specifying the `width` and `height` measurements in pixels.
- You can use whole and decimal values for percentages. For example, the values "4%" and "4.5%" are both valid.
- Because a region is clipped at the boundary of the root-layout region, no percentage value can effectively be more than 100%.

Adding a Background Color

By default, the root-layout region is black. All other regions use transparency as their default. In the SMIL layout, you can specify another background color for any region:

```
<layout>
  <root-layout background-color="maroon"/>
  <region id="videoregion" background-color="silver".../>
  <region id="textregion" background-color="#C2EBD7".../>
</layout>
```

For the color value, use any RGB hexadecimal value (#RRGGBB) supported by HTML, or one of the following predefined color names, listed here with their corresponding hexadecimal 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)

Using Transparency

A region is transparent if you do not define its background color. You can also specify "transparent" as a region background color. Transparency means that the region is not visible until a clip starts to play in it. However, this is not true transparency. If the media clip contains transparency too, you will not see through the clip and region to the root-layout region.

Fitting Clips to Regions

When a media clip is encoded at a size different from the playback region's defined size, the fit attribute determines how the clip fits the region:

```
<region id="videoregion" width="128" height="64" fit="meet"/>
```

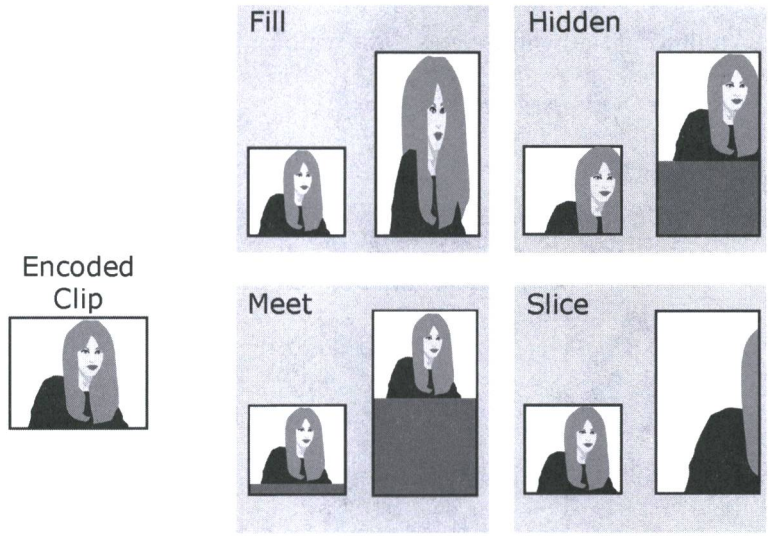
The fit attribute uses one of the values described in the following table. If you do not specify a fit attribute, the clip uses the default value `hidden`. In no case will media display outside the playback region's boundaries.

Region Fit Attributes

Attribute	Action
`fill`	Scale the clip so that it fills the region exactly. Image distortion occurs if the encoded clip and playback region have different height-to-width ratios.
`hidden` (default)	Keep the clip at its encoded size and place it at the region's upper, left-hand corner. If the clip is smaller than the region, fill remaining space with the region's background color. If the clip is larger than the region, crop out the area that doesn't fit.
`meet`	Place the clip at the region's upper, left-hand corner. Scale the clip and preserve its height/width ratio until one dimension is equal to the region's size and the other dimension is within the region's boundaries. Fill empty space with the region's background color.
`slice`	Place the clip at the region's upper, left-hand corner. Scale the clip and preserve its height/width ratio until one dimension is equal to the region's size and the other dimension overflows the region's boundaries. Crop the overflow.

The following figure illustrates the effect that fit attributes have on a source clip that plays in windows with different sizes and aspect ratios.

Different Clip Scaling Results Based on "fit" Attribute



Tip

When scaling media inside a region, keep in mind that different types of content scale with different results. A video scaled larger than its encoded size may not look good. Vector-based media such as RealFlash animation scale more easily to fit different region sizes, however. Also, scaling a clip consumes CPU on the RealPlayer machine.

Ordering Overlapping Regions with z-index

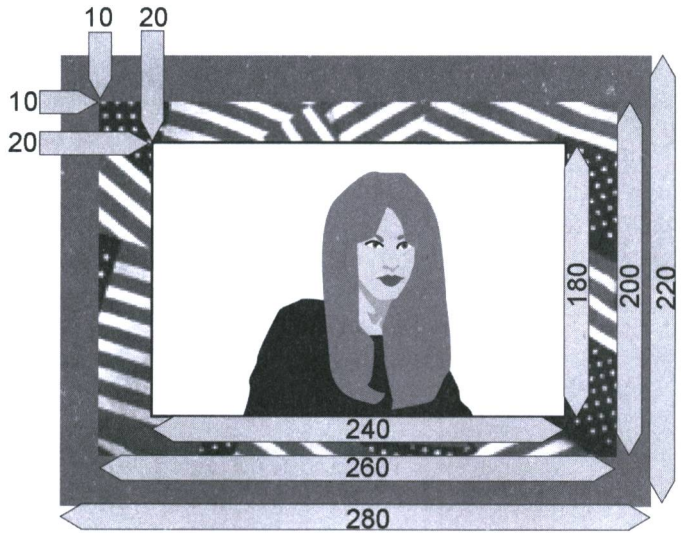
If regions overlap, you can use the `z-index` attribute to determine which regions appear in front. The following example creates a video region that overlaps an image region:

```
<layout>
  <root-layout background-color="gray" width="280" height="220"/>
  <region id="image" top="10" left="10" width="260" height="200" z-index="0"/>
  <region id="video" top="20" left="20" width="240" height="180" z-index="1"/>
</layout>
```

This example defines a gray root-layout region 220 pixels high by 280 pixels wide. A smaller image region is centered within this gray background. Its `z-index` value of zero makes it display behind all other regions, but not behind the root-layout region. The video region centered in the image region appears

on top of that region because of its higher value for z-index. Another region could overlap the video region with z-index set, for instance, to 2, 5, or 29. The following figure illustrates these regions.

Regions Overlapping through z-index



Tips for Defining Z-Index Values

The following are points to observe when using z-index:

- The root-layout region is always behind all other regions and does not use z-index.
- The z-index values can include negative integers (such as -4), 0 (zero), and positive integers (such as 5). A region with a z-index value of -4, for example, displays behind a region with a value of 0, which displays behind a region with a value of 5.
- The default value of 0 (zero) applies if you don't specify z-index.
- Using strictly sequential values such as 0, 1, 2, 3, 4 helps you keep track of the layers, but is not necessary. A sequence such as 0, 1, 6, 19, 34 works just as well, and leaving gaps in the sequence makes it easier to insert layers later.
- Nonoverlapping clips can have the same values. Side-by-side videos can both have z-index="3", for example.

- When overlapping clips have the same value, the clip that starts later in the presentation appears in front.

Additional Information

See "Hiding Regions with z-index" on page 174 for information on using z-index to make regions appear to come and go dynamically.

Assigning Clips to Regions

After you define the layout in the header section as described in"Defining the Layout" starting on page 96, you use region attributes within source tags to associate each clip with a region. In the following example, the video and text clips are assigned to the video and text regions defined in the header:

```
<smil>
 <head>
  <layout>
   <root-layout background-color="maroon" width="250" height="230"/>
   <region id="videoregion" top="5" left="5" width="240" height="180"/>
   <region id="textregion" top="200" left="5" width="240" height="20"/>
  </layout>
 </head>
 <body>
  <par>
   <video src="video.rm" region="videoregion"/>
   <audio src="audio.rm"/>
   <textstream src="text.rt" region="textregion"/>
  </par>
 </body>
</smil>
```

You can reuse regions by assigning sequential clips to them. For example, you can play a video clip in a region, then display another clip in that region after the first clip finishes. Don't assign the same region to two clips that play at the same time, however. You don't assign audio clips to regions at all because audio does not require a display region.

SMIL Layout Example

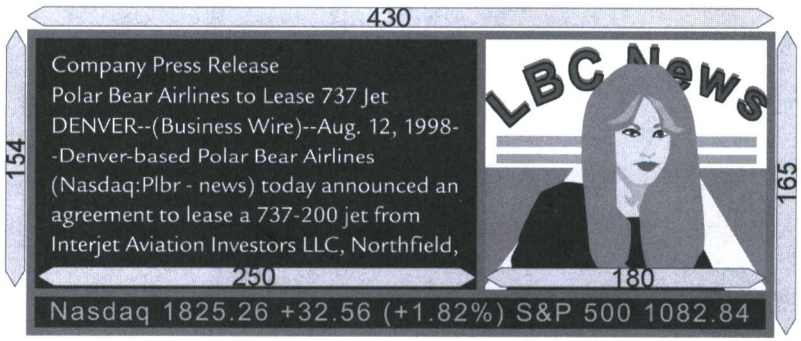
The following example displays three regions: a news region, a video region, and a stock ticker region. The news and video regions are arranged side by side

at the top of the RealPlayer display window. The stock ticker region appears below them.

```
<smil>
 <head>
  <!--presentation with 2 text clips and 1 video clip-->
  <meta name="title" content="Music of the Week"/>
  <layout>
   <root-layout width="430" height="165"/>
   <region id="newsregion" top="0" left="0" width="250" height="144"/>
   <region id="videoregion" top="0" left="250" width="180" height="144"/>
   <region id="stockregion" top="145" left="0" width="430" height="20"/>
  </layout>
 </head>
 <body>
  <par>
   <!--play these 3 clips simultaneously-->
   <textstream src="news.rt" region="newsregion"/>
   <video src="newsvid.rm" region="videoregion"/>
   <textstream src="stocks.rt" region="stockregion"/>
  </par>
 </body>
</smil>
```

The following figure illustrates the design of these regions.

SMIL Sample Layout with Text, Video, and Stock Ticker Windows



Linking to Other Media

A SMIL file can define links to other media. A video might link to a second video, for example. When the viewer clicks the link, the second video replaces the first. Or the video could link to an HTML page that opens in the viewer's

browser. You can even define areas as hot spots with links that vary over time. The bottom corner of a video can link to a different URL every ten seconds, for instance.

Note

Some media clips can also define hyperlinks. A RealText clip, for example, can define hyperlinks for portions of text. When a viewer clicks an area where a media clip link and a SMIL file link overlap, the SMIL link is used.

Making a Source Clip a Link

The simplest type of link connects an entire media source clip to another clip. As in HTML, you define the link with <a> and tags. But whereas you enclose text between <a> and in HTML, you enclose a media source tag between <a> and in SMIL:

```
<a href="rtsp://realserver.company.com/video2.rm">
  <video src="video.rm" region="videoregion"/>
</a>
```

The example above links the source clip video.rm to the target clip video2.rm. When a viewer moves the cursor over the source clip as it plays, the cursor turns to a hand icon to indicate that the clip is a link. When the viewer clicks video.rm as it plays, video2.rm replaces it. The URL begins with rtsp:// if the linked clip streams to RealPlayer from RealServer, or http:// if the file downloads to the browser from a Web server. When targeting a browser, be sure to include the show attribute as described below.

Additional Information

For information on RTSP URLs, see "Linking to Clips on RealServer" on page 78.

Targeting RealPlayer or a Browser

An <a> tag or <anchor> tag (see "Defining Hot Spot Links" on page 106) can include a show attribute that determines where a linked clip displays:

```
<a href="http://www.company.com/index.htm" show="new">
  <video src="video.rm" region="videoregion"/>
</a>
```

replace

The default attribute `show="replace"` causes the linked clip to replace the source clip in RealPlayer. This default behavior also occurs if you do not include the `show` attribute in the link. The following are important differences between RealPlayer and Web browsers to keep in mind when creating links:

- RealPlayer does not include a **Back** button that allows the viewer to return to the link source clip after clicking the link.
- Only one instance of RealPlayer can run at a time. You therefore cannot open a SMIL link in a new RealPlayer window the way you can open an HTML link in a new browser window.
- Clicking a link to another SMIL file or clip removes any existing regions. If you have three regions defined and the viewer clicks a link in one region, for example, the target clip replaces all media clips in all regions. You can preserve regions by targeting a SMIL file that defines the same set of regions. You cannot preserve the timeline positions of clips playing in those regions when the viewer clicks the link, however.

Additional Information

"Linking to a SMIL File" on page 109.

new, pause

The values `new` and `pause` both open the linked clip in the viewer's default browser. The source clip continues to play in RealPlayer if you use `show="new"`. With `show="pause"`, the source clip pauses in RealPlayer. The viewer can restart playback at any time, though, by clicking RealPlayer's **Play** button.

Use either `show="new"` or `show="pause"` to open a Web page or another clip viewable within a browser. You can use these attributes to link a RealSystem presentation to your home page, for example. Do not use them to link to another media clip played in RealPlayer, however, such as a SMIL file or a RealVideo clip.

Defining Hot Spot Links

Within a SMIL file you can define hot spots using an `<anchor>` tag. Whereas the `<a>` tag turns the entire media source clip into a link, the `<anchor>` tag turns only a defined area into a link. With `<anchor>` tags you can create links similar to those in HTML image maps. But SMIL links can be temporal as well as spatial. A link might be valid for just ten seconds during a source clip's timeline, for instance.

Setting an Anchor

The <anchor> tag differs from the <a> tag in that you place it within the media source tag rather than before it:

```
<video src="video.rm" region="videoregion">
  <anchor href="rtsp://realserver.company.com/video2.rm" .../>
</video>
```

An <anchor> tag ends with a closing slash. But the media source tag does not end with a closing slash as it normally would. Instead, the source tag and its subsequent <anchor> tags are followed by a closing source tag, such as </video>. The <anchor> tag includes an href attribute that uses rtsp:// if the linked clip streams from RealServer, or http:// if the file downloads to the browser from a Web server

Additional Information

For information on RTSP URLs, see "Linking to Clips on RealServer" on page 78. To target a browser with a link, see "Targeting RealPlayer or a Browser" on page 105.

Defining Spatial Coordinates

The <anchor> tag's coords attribute defines spatial coordinates for the hot spot rectangle. Coordinate values in pixels or percentages define the rectangle's offset from the upper, left-hand corner of the media source clip as shown in this example:

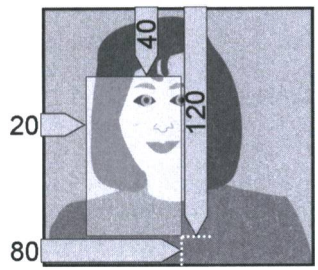
```
<video src="video.rm" region="videoregion">
  <anchor href="..." coords="20,40,80,120"/>
</video>
```

The coordinate values for the hot spot rectangle follow this order:

1. left side pixel or percentage value
2. top pixel or percentage value
3. right side pixel or percentage value
4. bottom pixel or percentage value

The sample above uses pixel values to define a hot spot 60 pixels wide (80 pixels minus 20 pixels) 80 pixels high (120 pixels minus 40 pixels). It produces a hot spot like that shown in the following figure.

Hot Spot (Hyperlink) Defined with SMIL



Percentage Values

The coords attribute can also use percentage values to create hot spots relative to the source clip's size. The following sample places in the center of the source clip a hot spot that is a quarter the size of the source clip:

```
<video src="video.rm" region="videoregion">
  <anchor href="..." coords="25%,25%,75%,75%"/>
</video>
```

The following table lists sample percentage coordinates that define hot spots for a source clip. Each hot spot is a quarter the size of the source clip.

Sample Hot Spot Percentage Coordinates

Hot Spot Rectangle Position	Coordinate Attribute
Upper, left-hand quadrant	coords="0,0,50%,50%"
Upper, right-hand quadrant	coords="50%,0,100%,50%"
Lower, left-hand quadrant	coords="0,50%,50%,100%"
Lower, right-hand quadrant	coords="50%,50%,100%,100%"
Center	coords="25%,25%,75%,75%"

Tips for Defining Anchor Coordinates

Note the following when defining hot spots:

- You can mix pixels and percentages. For example, the coordinates "50,50,100%,100%" place the hot spot's left and top boundaries in and down 50 pixels from the source clip's upper, left-hand corner, respectively. But the hot spot's right and bottom boundaries extend to the source clip's right and bottom edges, respectively, no matter the source clip's size.
- Values such as "30,30,10,10" are ignored. Here, the hot spot's left side is defined as farther to the right than its right side. As well, the top is defined to be below the bottom.

- You can use whole and decimal values for percentages. For example, the values "4%" and "4.5%" are both valid.
- A hot spot defined to extend beyond the source clip is clipped at the clip's edge. For example, if the hot spot has coordinates "50,50,300,300" but the source clip is 200 by 200 pixels, the hot spot's effective coordinates are "50,50,200,200". For this reason, no percentage value can effectively be more than 100%.

Setting Temporal Coordinates

In addition to defining spatial coordinates, the <anchor> tag can set temporal attributes that specify when the link is active. If you do not include temporal attributes, the link stays active as long as the source clip appears on screen. To add timing attributes, use the SMIL begin and end values. (You cannot use dur, clip-begin, or clip-end.)

The following example creates two temporal links for the clip video.rm. The first link is active for the first five seconds of playback. The second link is active for the next five seconds. Because no spatial coordinates are given, the entire video is a link:

```
<video src="video.rm" region="videoregion">
  <anchor href="rtsp://.../video2.rm" begin="0s" end="5s"/>
  <anchor href="rtsp://.../video3.rm" begin="5s" end="10s"/>
</video>
```

Additional Information

See "Setting Begin and End Times" on page 85. The attributes use the SMIL timing values described in "Specifying Timing" on page 84.

Linking to a SMIL File

A SMIL file can define a link to another SMIL file or another part of itself. For example, a video played through a SMIL file may link to another SMIL file so that when a viewer clicks the video, a new presentation starts up in RealPlayer. To do this, you simply set the href attribute for the <a> or <anchor> tag to the new SMIL file's URL.

You can also link to portions of a SMIL file. The following example from a target SMIL file uses id attributes (such as those used in regions to create region names) to define a target name for a <par> tag that groups a video and a text clip. This id attribute functions like a name attribute in an HTML <a> tag:

```
<par id="text_and_video">
  <video src="video2.rm" region="newsregion"/>
  <textstream src="text.rt" region="textregion"/>
</par>
```

You then link the source SMIL file to the named target by including a pound sign ("#") and the target name within the link URL. Assuming the target SMIL file is named newmedia.smil, the source file's link to the <par> group looks like this:

```
<a href="rtsp://realserver.company.com/newmedia.smil#text_and_video">
  <video src="video.rm" region="videoregion"/>
</a>
```

Note that the target SMIL file defines two regions, newsregion and textregion. When RealPlayer receives the new SMIL file, it creates those regions as specified in the file's header.

Tips for Linking to a SMIL File

Note the following when linking to another SMIL file:

- You can link to any media clip or <par> or <seq> group by defining an id attribute for the clip or group. Do not link to an element in a SMIL file header, however. Although you can link to a <switch> group, do not link to a clip or group within a <switch> group.
- You cannot link to a media clip in a <par> group and exclude the other clips in that group. All clips in the group will play in their designated regions.
- If additional clips follow the target clip in the SMIL file, those clips play when the target clip finishes playback.

  Tip

  If you want to link to a single clip but the SMIL file that refers to that clip lists other clips as well, link to the clip directly. Or create a new SMIL file that lists only the single target clip.

- To link to a target within the same SMIL file, simply set the href attribute value to the target id, such as <a href="#text_and_video">. Be sure to include the pound sign before the id value.

Linking with a Timeline Offset

You can use the <anchor> tag's time coordinates to create a timeline offset in a linked clip. Suppose you want to link a video to another video at 30 seconds into the second video's timeline. In the source SMIL file, you define an <a> or <anchor> link from the first video to a SMIL file that contains the second video. In the second SMIL file, the video's <anchor> tag defines the timeline offset using SMIL timing parameters.

Here is a sample of the link in the first SMIL file:

```
<a href="rtsp://realserver.company.com/newmedia.smil#vid2">
  <video src="video.rm" region="videoregion"/>
</a>
```

The following is the linked video clip in the second SMIL file, newmedia.smil:

```
<video src="video2.rm" region="newsregion">
  <anchor id="vid2" begin="30s"/>
</video>
```

Additional Information

"Specifying Timing" on page 84 describes the SMIL timing values.

Defining Image Options

For a still image, you can use the following options to modify the image's streaming characteristics, or to link it to RealPlayer commands. These options are not SMIL parameters, but extensions to the image's SMIL source tag, which is described in "Specifying Clip Locations" on page 77. The following table summarizes the image options, which are described in detail below.

Options for Image Source Tag

Option	Function	Format	Values	Example
bitrate	Sets streaming bit rate.	all	bits per second	bitrate=1000
bgcolor	Substitutes color for transparency.	GIF	color name or hex value	bgcolor=black bgcolor=AA3344
url	Links to URL or a RealPlayer command.	all	URL or command	url=http://www.real.com url=command:stop()

(Table Page 1 of 2)

Options for Image Source Tag (continued)

Option	Function	Format	Values	Example
target	Sets browser or RealPlayer as target.	all	_browser _player	target=_browser target=_player
reliable	Ensures image arrival.	all	true	reliable=true

(Table Page 2 of 2)

A question mark operator ("?") separates image options from the image URL in the SMIL source tag. To use multiple image options, put a question mark before the first option, then separate the remaining options with ampersands ("&"). The order of options does not matter. The following shows the general form for two image options added to the image source tag. Note that image options and values are not in quotation marks because they are part of the quoted src value:

```
<image src="URL?option=value&option=value"/>
```

Here is an example that combines bit rate and background color options:

```
<image src="button.gif?bitrate=1000&bgcolor=blue"/>
```

Setting Image Bit Rates

By default, an image streams at 12 Kbps until RealPlayer receives it. You can set a higher bit rate to take advantage of bandwidth availability and stream an image quickly. Or you can set a lower bit rate when streaming images in parallel with another clip so that the other clip has enough bandwidth for uninterrupted playback. The following example shows three sequential image files set to stream at a low bit rate to ensure that a video playing in parallel does not stall:

```
<par>
 <video src="video.rm" region ="videoregion"/>
 <par>
  <seq>
   <image src="ad1.gif?bitrate=1000" begin="30s" dur="60s" region="adregion"/>
   <image src="ad2.gif?bitrate=1000" dur="120s" region="adregion"/>
   <image src="ad3.gif?bitrate=1000" dur="60s" region="adregion"/>
  </seq>
 </par>
</par>
```

In this example, each GIF image is set to stream at 1 Kbps. This ensures that each image consumes a small amount of bandwidth and does not interfere with video playback. The <par> tag just outside the <seq> tag makes RealServer

balance the bit rates of the images as a single group rather than as a discrete sequence.

Additional Information

For more on using <par> to balance bandwidth, see "Smoothing Transitions between Clips" on page 173.

Keep in mind that the image size divided by the maximum bit rate equals the minimum time for streaming the image. Assuming images in this example are 30 Kilobits, each image takes about 30 seconds to stream. For this reason, the first image is set to display 30 seconds after the video begins to play. If no begin time were defined, the video would be delayed up to 30 seconds while the first image streams to RealPlayer. As well, each successive image must be able to stream during the preceding image's duration.

Overriding GIF Transparency

For GIF images that include transparency, you can use bgcolor to substitute a color for the transparency. Color values are any hexadecimal color value without the leading pound sign ("#"), or a predefined color name as described in "Adding a Background Color" on page 99. If you do not set the color, the region's defined background color shows through transparent spots in the image. Here is an example:

```
<image src="button.gif?bgcolor=BB21AA"/>
```

Linking to a Web Page or Clip

The image url and target options let you link an image with a fully qualified URL. When the viewer clicks the image, the URL opens in the viewer's browser or RealPlayer. The following example shows an image linked with a URL that opens in the viewer's default browser:

```
<image src="home.gif?url=http://www.company.com&target=_browser" region="home"/>
```

The next example shows an image linked with a URL that opens in RealPlayer:

```
<image src="go.gif?url=rtsp://realserver.company.com/media.smil&target=_player"
 region="next"/>
```

Keep in mind that opening a URL in RealPlayer replaces the current presentation with the new clip or SMIL file.

Note

The image hyperlink options duplicate the SMIL hyperlink features described in "Linking to Other Media", beginning on page 104. You can use either method depending on your preference.

Controlling RealPlayer

With url=command:*command*&target=_player, you can make the image a link that causes RealPlayer to stop, pause, play, or seek to a specific point in the presentation timeline. The following example shows three images set to appear in three separate SMIL regions. Each image issues a different RealPlayer command when clicked:

```
<par>
  <image src="play.gif?url=command:play()&target=_player" region="play"/>
  <image src="pause.gif?url=command:pause()&target=_player" region="pause"/>
  <image src="stop.gif?url=command:stop()&target=_player" region="stop"/>
</par>
```

An image can also issue a seek command that specifies a certain point in the presentation timeline:

```
<image src="seek.gif?url=command:seek(1:35.4)&target=_player"
  region="seek"/>
```

In the example above, clicking the image instructs RealPlayer to seek to 1:35.4 into the presentation. This option does not use SMIL shorthand timing values such as s for seconds or min for minutes. Instead, the time format is like the "Normal Play Time Format" described on page 85, except that it also accepts milliseconds and days as possible values.

```
dd:hh:mm:ss.xyz
```

Here, dd is days, hh is hours, mm is minutes, ss is seconds, x is tenths of seconds, y is hundredths of seconds, and z is milliseconds. Only the ss field is required.

When the time value does not include a decimal point, the last field is read as the seconds. For example, 1:30 means 1 minute and 30 seconds, whereas 1:30:00 means 1 hour and 30 minutes. Thus all the following values seek to the point in the timeline 90 minutes after the presentation begins:

```
url=command:seek(1:30:00.0)&target=_player
url=command:seek(90:00)&target=_player
url=command:seek(5400)&target=_player
```

Ensuring Reliable Image Transmission

Use the reliable=true option to indicate that the image must be delivered to RealPlayer under any circumstance:

```
<image src="button.gif?reliable=true"/>
```

During extremely adverse network conditions, RealSystem will halt the presentation if necessary rather than drop the image. You should use this option sparingly, though, because RealSystem normally ensures that very little data loss occurs in transmission.

PLAYING CLIPS IN A WEB PAGE



Embedding a presentation in a Web page provides seamless playback without launching RealPlayer as a separate application. You can even include RealPlayer controls such as fast forward and pause. This chapter explains how you add mark-up to a Web page so that people view your RealSystem G2 presentation directly through their Web browsers.

Additional Information

Refer to this chapter for full information on embedding a presentation. For a quick example of how to embed a video in a Web page, see "Embedding a RealVideo Clip in a Web Page" on page 169.

Choosing the Netscape Plug-in or ActiveX Control

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

- Netscape Navigator 3.0, 4.0, and higher.
- Microsoft Internet Explorer 3.0 and 4.0.

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

- Microsoft Internet Explorer 3.0 and 4.0.
- Any application that supports ActiveX controls, such as Visual Basic, Visual C++, Microsoft Access, and so on.

Because they both have the same capabilities, you can use either the plug-in or the ActiveX control depending on which products you need to support. The following sections describe the basics of using the plug-in or the control, then explain each parameter you can set.

Note

Familiarity with RealPlayer and HTML will make it easier to use the plug-in or control. In the examples below, ellipses (...) indicate information left out to keep the examples short. Make sure your tags include all necessary information.

Additional Information

Embedded RealPlayer Extended Functionality Guide at **http://service.real.com/help/library/index.html** explains how to use JavaScript or VBScript to extend the functionality of the Netscape Plug-in or ActiveX control.

Using <EMBED> Tags for the Netscape Plug-In

To use RealPlayer's Netscape plug-in, you add <EMBED> tags to your Web page HTML. Each <EMBED> tag has three required parameters (SRC, WIDTH, HEIGHT), and can include many optional parameters. The basic <EMBED> tag looks like the following (the SRC value, which is explained below, has been omitted for simplicity):

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

This tag creates a playback area 300 pixels wide by 134 pixels high within the Web page. Parameters typically have the form PARAMETER=value. The parameter names can be any case, though this manual shows them 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.

Supporting Other Browsers

To accommodate browsers that do not support the Netscape plug-in, use <NOEMBED> to define a standard hypertext link to your presentation. The unembedded link follows the <EMBED> tag:

```
<EMBED SRC="..." WIDTH=300 HEIGHT=134>
<NOEMBED><A HREF="...">Play with RealPlayer.</A></NOEMBED>
```

In this example, browsers that can play the embedded presentation hide the text between <NOEMBED> and </NOEMBED>. Other browsers ignore the preceding <EMBED> tag and display just the hypertext link. The user then clicks the link to play the presentation in RealPlayer.

Using <OBJECT> Tags for the ActiveX Control

You embed the RealPlayer ActiveX control in HTML pages with the <OBJECT> tag. This tag uses an ID that you select, such as ID=RVOCX, and must have the following class ID that identifies RealPlayer:

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

The <OBJECT> tag also sets the width and height of the playback area within the browser. A typical <OBJECT> tag looks like this:

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

This tag creates a playback area 300 pixels wide by 134 pixels high within the Web page. Between <OBJECT> and </OBJECT>, you can define any number of additional parameters in this form:

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

PARAM, NAME, and VALUE markers can be any case, though this manual shows them uppercase. Parameter values are not case-sensitive except for file names, which must typically be lowercase. Always enclose parameter values in double quotation marks.

Setting Basic Parameters

Both the Netscape plug-in and ActiveX control use the same basic tag parameters. As explained above, the tag syntax for the plug-in and the control differs, however. The following sections explain the basic parameters you can include in each <EMBED> or <OBJECT> tag.

SRC

The SRC parameter gives the presentation's source URL surrounded by double quotes. The directory names cannot contain spaces. For the ActiveX control, the <OBJECT> tag's CLASSID parameter causes the presentation to play back in the Web page, so you can simply link to the SMIL file or clip within one <OBJECT> tag. When linking to a presentation on RealServer, the URL can use Ramgen, but this is not necessary:

```
<PARAM NAME="SRC" VALUE="http://realserver.company.com:8080/ramgen/sample.smil">
```

When you use the Netscape plug-in, you need to include the SRC parameter in every <EMBED> tag. You can specify two types of URLs. The first uses the

?embed option to RealServer's Ramgen, which causes RealPlayer to play the clip or SMIL presentation through its Netscape plug-in instead of launching as a separate application:

```
SRC="http://realserver.company.com:8080/ramgen/sample.smil?embed"
```

Additional Information

For more on Ramgen and presentation URLs, see "Linking your Web Page to RealServer" on page 147.

The second option is to create a Ram file with a .rpm extension and list this file in the SRC parameter without including Ramgen and the ?embed option. The Ram file then links to your SMIL file or clips. The following example links to a .rpm file on a Web server:

```
SRC="http://www.company.com/media/sample.rpm"
```

Additional Information

For instructions on writing a .rpm file, see "Creating a Ram File Manually" on page 153.

You must create and link to a .rpm file in any of the following situations:

- Your RealServer does not use Ramgen.
- The SMIL file or clip is on a Web server.
- All files are located on a user's local computer rather than streamed over a network.

Note

When linking to local files, you can use relative links or absolute links. When you use absolute links with the Netscape plug-in, though, use forward slashes in paths. For example, instead of file://c:\media\first.rpm, set the source as file://c:/media/first.rpm.

WIDTH and HEIGHT

Required for each <EMBED> or <OBJECT> tag, the WIDTH and HEIGHT parameters set the size of the playback area. If you leave them out, the playback area may appear as a tiny icon because streaming media presentations do not size automatically. The values for WIDTH and HEIGHT are in pixels by default, so a width of 300 creates a playback area 300 pixels wide. Setting WIDTH and HEIGHT to 0 (zero) hides the playback area.

You can also express WIDTH and HEIGHT as percentages of the browser window size. For example, a width of 50% makes the presentation area half the browser width. Keep in mind that different types of media scale with different results. A video scaled larger than its encoded size may not look good. Vector-based media such as RealFlash animation scale more easily to fit different playback areas, however.

NOJAVA

Setting NOJAVA to true in every <EMBED> tag prevents the browser's Java Virtual Machine (JVM) from starting if it is not yet running:

```
<EMBED SRC="..." WIDTH=300 HEIGHT=134 NOJAVA=true>
```

This is recommended when using the Netscape plug-in because the JVM is not required for the <EMBED> parameters described in this chapter. Starting the JVM therefore delays presentation playback unnecessarily. The JVM is required only when extending plug-in functionality with JavaScript. In this case, omit NOJAVA entirely from the <EMBED> tags.

Additional Information

See *Embedded RealPlayer Extended Functionality Guide* at **http://service.real.com/help/library/index.html** for more information on controlling RealPlayer with JavaScript.

Note

Although you can include NOJAVA as an ActiveX parameter, it has no effect on Internet Explorer, which launches its JVM on browser start-up.

Adding RealPlayer Controls

With the CONTROLS parameter, you can place RealPlayer controls such as a play/pause button in your Web page. A visitor to your page can then control the presentation playback just as if using RealPlayer as a separate application. The following example for the Netscape <EMBED> tag displays the play/pause button:

```
<EMBED SRC="..." WIDTH=26 HEIGHT=26 NOJAVA=true CONTROLS=PlayButton>
```

For the ActiveX control, you define a CONTROLS parameter within the <OBJECT> tag structure:

```
<OBJECT ID=RVOCX CLASSID="..." WIDTH=26 HEIGHT=26>
<PARAM NAME="SRC" VALUE="...">
<PARAM NAME="CONTROLS" VALUE="PlayButton">
</OBJECT>
```

The following sections describe each RealPlayer control you can use. A tag's `WIDTH` and `HEIGHT` parameters set the control's size. The suggested pixel widths and heights given below produce embedded controls approximately the same sizes as the RealPlayer controls. Specifying different pixel sizes scales the controls larger or smaller. You can also use percentage values for sizes, but this is recommended only for the image window.

Additional Information

For information on adding more than one control to your Web page, see "Linking Multiple Controls" on page 127.

Tip

Download the HTML version of this manual from **http://service.real.com/help/library/index.html** to see online examples of embedded RealPlayer controls.

Basic Controls

ImageWindow



Displays the image window. This is available only for display presentations such as video or animation. Even if no other controls are visible on the page, the user can typically right-click (on Windows) or hold down the mouse button (Macintosh) on the playback area to display a menu of choices such as **Play** and **Stop**. See also "Controlling Image Display" on page 129.

Suggested pixel width: 176 or higher

Suggested pixel height: 132 or higher

All



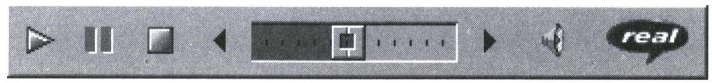
Displays the basic RealPlayer control panel. The control name "default" also works. Buttons include play, pause, stop, fast forward, and rewind. Sliders include a position slider and a volume slider with mute button that pops up when the speaker button is clicked. Below the buttons are a clip information field, a status panel, a network congestion indicator, and a clip timing field.

Suggested pixel width: 375

Suggested pixel height: 100

Individual Controls and Sliders

ControlPanel



Displays the play, pause, stop, fast forward and rewind buttons. 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

PlayButton



Displays a play/pause button.

Suggested pixel width: 44

Suggested pixel height: 26

PlayOnlyButton



Displays a play button.

Suggested pixel width: 26

Suggested pixel height: 26



PauseButton

Displays a pause button.

Suggested pixel width: 26

Suggested pixel height: 26



StopButton

Displays a stop button.

Suggested pixel width: 26

Suggested pixel height: 26



FFCtrl

Displays a fast forward control.

Suggested pixel width: 26

Suggested pixel height: 26



RWCtrl

Displays a rewind control.

Suggested pixel width: 26

Suggested pixel height: 26



MuteCtrl

Displays a mute button.

Suggested pixel width: 26

Suggested pixel height: 26



MuteVolume

Displays a mute button and volume slider.

Suggested pixel width: 26

Suggested pixel height: 88



VolumeSlider

Displays a volume slider.

Suggested pixel width: 26

Suggested pixel height: 65

PositionSlider



Displays a clip position slider.

Suggested pixel width: 120

Suggested pixel height: 26

TACCtrl



Displays a clip information field. For instructions on formatting the clip playlist, see "Managing the Playlist" on page 92.

Suggested pixel width: 370

Suggested pixel height: 32

HomeCtrl



Displays the Real™ logo.

Suggested pixel width: 45

Suggested pixel height: 25

Information Panels

InfoVolumePanel



Displays presentation information along with the volume slider and mute button. For more on presentation information, see "Adding Presentation Information" on page 91. See also "NOLABELS" on page 130.

Suggested pixel width: 325

Suggested pixel height: 55

InfoPanel



Displays the presentation information panel. For more on presentation information, see "Adding Presentation Information" on page 91. See also "NOLABELS" on page 130.

Suggested pixel width: 300

Suggested pixel height: 55

Status Panels

StatusBar



Displays the status panel, which shows informational messages. It also includes the network congestion LED and the position field, which shows current place in the presentation timeline along with total clip length.

Suggested pixel width: 300

Suggested pixel height: 30

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



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

PositionField



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

Suggested pixel width: 90

Suggested pixel height: 30

Linking Multiple Controls

The CONSOLE parameter defines a name that unifies <EMBED> or <OBJECT> tags so that multiple controls work together. For example, you could create three separate <EMBED> or <OBJECT> tags to define an image window, a play button, and a stop button. By using three tags, you can set the size of each control separately and define its layout with HTML tags. You could put each control in a different HTML table cell, for example.

To tie controls together, define the same CONSOLE name within each <EMBED> or <OBJECT> tag, or use one of these predefined names:

_master links the control to all other embedded controls on the page.

_unique links the control to no other embedded controls on the page.

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

Tips for Using Consoles

Note the following when grouping multiple controls with CONSOLE attributes:

- Every <EMBED> tag must have a SRC attribute. Tags linked by a console name should have the same SRC value.
- With the ActiveX control, only one <OBJECT> tag in a console group needs to have a SRC value.
- If the <EMBED> or <OBJECT> tags in a console group have different SRC values, the first valid source that RealPlayer 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 examples for the <EMBED> and <OBJECT> tags set up an image window and two sets of controls (a play button and stop button) for separate videos, sample1.rm and sample2.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.

Netscape Plug-in Sample Mark-up

Because each <EMBED> tag must have a SRC value, the image window in the following example simply 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 video.

```
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample1.rm?embed"
WIDTH=176 HEIGHT=128 NOJAVA=true CONTROLS=ImageWindow CONSOLE=_master>

<H4>Video 1</H4>
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample1.rm?embed"
WIDTH=44 HEIGHT=26 NOJAVA=true CONTROLS=PlayButton CONSOLE=video1>
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample1.rm?embed"
WIDTH=26 HEIGHT=26 NOJAVA=true CONTROLS=StopButton CONSOLE=video1>

<H4>Video 2</H4>
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample2.rm?embed"
WIDTH=44 HEIGHT=26 NOJAVA=true CONTROLS=PlayButton CONSOLE=video2>
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample2.rm?embed"
WIDTH=26 HEIGHT=26 NOJAVA=true CONTROLS=StopButton CONSOLE=video2>
```

ActiveX Control Sample Mark-up

In the following ActiveX example, only the controls for the two play buttons define the source URLs. For convenience the CLASSID value is omitted.

```
<OBJECT ID=RVOCX CLASSID="..." WIDTH=176 HEIGHT=128>
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="CONSOLE" VALUE="_master">
</OBJECT>
```

```
<H4>Video 1</H4>
<OBJECT ID=RVOCX CLASSID="..." WIDTH=44 HEIGHT=26>
<PARAM NAME="SRC"
 VALUE="http://realserver.company.com:8080/ramgen/sample1.rm">
<PARAM NAME="CONTROLS" VALUE="PlayButton">
<PARAM NAME="CONSOLE" VALUE="video1">
</OBJECT>
<OBJECT ID=RVOCX CLASSID="..." WIDTH=26 HEIGHT=26>
<PARAM NAME="CONTROLS" VALUE="StopButton">
<PARAM NAME="CONSOLE" VALUE="video1">
</OBJECT>

<H4>Video 2</H4>
<OBJECT ID=RVOCX CLASSID="..." WIDTH=44 HEIGHT=26>
<PARAM NAME="SRC"
 VALUE="http://realserver.company.com:8080/ramgen/sample2.rm">
<PARAM NAME="CONTROLS" VALUE="PlayButton">
<PARAM NAME="CONSOLE" VALUE="video2">
</OBJECT>
<OBJECT ID=RVOCX CLASSID="..." WIDTH=26 HEIGHT=26>
<PARAM NAME="CONTROLS" VALUE="StopButton">
<PARAM NAME="CONSOLE" VALUE="video2">
</OBJECT>
```

Controlling Image Display

The following <EMBED> and <OBJECT> parameters control aspects of how clips play back.

BACKGROUNDCOLOR

This parameter specifies a background color for the image window. The specified background color also shows through if a clip includes transparency. The background color is black by default. You can use an RGB hexadecimal color value (#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)

Note

SMIL region background colors override this background color. For more on setting SMIL region colors, see "Adding a Background Color" on page 99.

CENTER

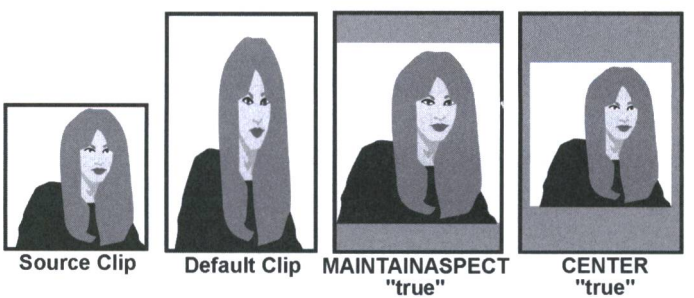
The default value for CENTER is false, which causes the clip to fill the entire playback area. If you set CENTER to true, the clip is centered within the playback area and displays at its encoded size. So by setting CENTER to true, you can create a large playback area with WIDTH and HEIGHT and still have the clip play at its normal size. You cannot use CENTER along with MAINTAINASPECT.

MAINTAINASPECT

This parameter, which you cannot combine with CENTER, determines whether the clip's height-to-width ratio stays constant when the clip scales to fit the image window. The default value of false causes this ratio to change as necessary to fill the image window fully. This may distort the source image.

If you set MAINTAINASPECT to true, a clip's height-to-width ratio stays constant. For example, a clip's height-to-width ratio of 1:1 stays constant even if the image window's height-to-width ratio is 2:3. In these cases, 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. The following figure shows how clips scale by default, with MAINTAINASPECT set to true, and with CENTER set to true.

Clip Scaling with MAINTAINASPECT and CENTER



NOLABELS

When you use a control that includes presentation information such as title, author, and copyright, you can include the NOLABELS option to suppress that

information in RealPlayer 5.0. This option has no effect on RealPlayer G2. Here is an example for the Netscape plug-in:

```
<EMBED ... CONTROLS=All NOLABELS=true>
```

and an example for the ActiveX control:

```
<OBJECT ...>
<PARAM NAME="CONTROLS" VALUE="All">
<PARAM NAME="NOLABELS" VALUE="true">
</OBJECT>
```

Additional Information

"Parameter Reference" on page 135 lists the options that work with RealPlayer 5.0.

NOLOGO

When set to true, NOLOGO prevents the Real logo from displaying in the image window. When no clip is playing, only the specified background color shows in the window. The parameter is false by default.

Setting Automatic Playback

The AUTOSTART, LOOP, and NUMLOOP parameters let you set the content to start playing automatically, as well as loop continuously for a specified number of times. The following example shows two of these parameters used with the Netscape plug-in:

```
<EMBED SRC="..." WIDTH=50% HEIGHT=50% NOJAVA=true AUTOSTART=true
LOOP=true>
```

Here is an example for the ActiveX control:

```
<OBJECT ID=RVOCX CLASSID="..." WIDTH=50% HEIGHT=50%>
<PARAM NAME="SRC" VALUE="...">
<PARAM NAME="AUTOSTART" VALUE="true">
<PARAM NAME="LOOP" VALUE="true">
</OBJECT>
```

AUTOSTART

When set to true, the AUTOSTART parameter starts playback immediately. When you have multiple <EMBED> or <OBJECT> tags linked by a CONSOLE name, set AUTOSTART to true in just one tag. Leaving AUTOSTART out or setting its value to false means the presentation does not start until the user starts it, for example, by clicking an embedded play button.

LOOP

If the LOOP parameter is set to true, the presentation continuously loops until the viewer stops it. When you have multiple <EMBED> or <OBJECT> tags linked by a CONSOLE name, set looping in just one tag. If you leave the LOOP parameter out, the default value of false applies and the presentation stops after the first playback. The user can play the presentation again by clicking the play button.

NUMLOOP

If you specify a NUMLOOP value such as "2", the presentation loops the specified number of times and then stops. If both LOOP and NUMLOOP are used, the LOOP parameter is ignored.

SHUFFLE

The SHUFFLE parameter is for use only with Ram or SMIL files that list a single sequence of clips. When set to true, SHUFFLE makes RealPlayer play back the clips in a random order.

Laying out SMIL Presentations

As "Laying Out Multiple Clips" on page 95 explains, you can use a SMIL file to define separate playback regions for different parts of a presentation. This lets you lay out two clips side-by-side, for example. When playing a presentation in a Web page, you can define the layout in SMIL or in HTML.

Defining the Layout with SMIL

To control the layout through SMIL, you set up the regions and their relative placements in the SMIL file. You then use the Netscape plug-in or ActiveX control to create a playback region in the Web page large enough to accommodate all regions. This SMIL file then produces the same layout when played through the Web page or RealPlayer.

The sample layout shown in "SMIL Layout Example" on page 103 defines three regions, creating a total playback area 430 pixels wide by 165 pixels high. To accommodate this in your Web page, you define an image window at least as large as this through the <EMBED> or <OBJECT> tag. Here are examples for the Netscape plug-in:

```
<EMBED SRC="..." WIDTH=430 HEIGHT=165 NOJAVA=true CONTROLS=ImageWindow
CONSOLE=one>
```

and ActiveX control:

```
<OBJECT ID=RVOCX CLASSID="..." WIDTH=430 HEIGHT=165>
<PARAM NAME="SRC" VALUE="...">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="CONSOLE" VALUE="one">
</OBJECT>
```

The SRC parameter provides the URL to the SMIL file. You can then use additional <EMBED> or <OBJECT> tags linked to the "one" console to provide RealPlayer controls for the presentation.

Defining the Layout with HTML

The second method omits SMIL layout information and defines the layout through HTML. You could place an image window and separate RealPlayer controls in an HTML table, for example. Each <EMBED> or <OBJECT> tags then uses a REGION parameter to define a region name. Clip source tags within the SMIL file list which region each clip plays in. For example, the SMIL file shown in "SMIL Layout Example" on page 103 lists the following RealText file, set to play in the newsregion region:

```
<textstream src="news.rt" region="newsregion"/>
```

Within the HTML page, the <EMBED> tag that plays news.rt would look like this:

```
<EMBED SRC="http://realserver.company.com:8080/ramgen/sample.smil?embed"
WIDTH=250 HEIGHT=144 NOJAVA=true CONTROLS=ImageWindow
REGION=newsregion CONSOLE=one>
```

The <OBJECT> tag would look like this:

```
<OBJECT ID=RVOCX CLASSID="..." WIDTH=250 HEIGHT=144>
<PARAM NAME="SRC"
  VALUE="http://realserver.company.com:8080/ramgen/sample.smil">
<PARAM NAME="CONTROLS" VALUE="ImageWindow">
<PARAM NAME="REGION" VALUE="newsregion">
<PARAM NAME="CONSOLE" VALUE="one">
</OBJECT>
```

You define similar <EMBED> or <OBJECT> tags to create other regions for other clips listed in the SMIL file. The SRC parameter in each tag lists the same SMIL file. You can also use additional <EMBED> or <OBJECT> tags linked to the same console to provide RealPlayer controls for the presentation.

Omitting SMIL Layout Information

When you define the presentation layout through HTML, you must omit the <layout> section from the SMIL file header. For example, the SMIL file shown in "SMIL Layout Example" on page 103 would not have a <layout> section:

```
<smil>
  <head>
    <!--presentation with 2 text clips and 1 video clip-->
    <meta name="title" content="Music of the Week"/>
  </head>
  <body>
    <par>
      <!-- play these 3 clips simultaneously -->
      <textstream src="news.rt" region="newsregion"/>
      <video src="newsvid.rm" region="videoregion"/>
      <textstream src="stocks.rt" region="stockregion"/>
    </par>
  </body>
</smil>
```

Tip

A SMIL file without a layout still works with RealPlayer G2, but RealPlayer automatically creates the layout, and the results may not be what you expect. Always include a SMIL layout when playing a presentation directly in RealPlayer.

Note

Because there is no SMIL layout, hot spot links are ignored. You can make media clips hyperlinks with the SMIL <a>... tags, however. For more on SMIL hyperlinks, see "Linking to Other Media" on page 104.

Parameter Reference

The following table lists the <EMBED> and <OBJECT> tag parameters. Parameters compatible with RealPlayer 5.0 are indicated in the "5.0?" column. To have an embedded presentation play back with RealPlayer 5.0 as well as RealPlayer G2, use only 5.0-compatible parameters.

<EMBED> and <OBJECT>Tag Parameter s

Parameter	Function	5.0?	Values	Default	Refer To
AUTOSTART	Sets automatic playback.	**yes**	true|false	false	page 131
BACKGROUNDCOLOR	Sets background color.	no	color name or RGB hex value	black	page 129
CENTER	Centers clip in window.	no	true|false	false	page 130
CONSOLE	Links multiple controls.	**yes**	name, _master, or _unique	(none)	page 127
CONTROLS	Adds RealPlayer controls.	**yes**	control name	All	page 121
HEIGHT	Sets window or control height.	**yes**	percentage or pixels	(none)	page 120
LOOP	Loops clips indefinitely.	no	true|false	false	page 132
MAINTAINASPECT	Preserves image aspect ratio.	no	true|false	false	page 130
NOJAVA	Prevents JVM start-up.	no	true|false	false	page 121
NOLABELS	Suppresses presentation information in RealPlayer 5.0.	**yes**	true|false	false	page 130
NOLOGO	Suppresses RealLogo.	no	true|false	false	page 131
NUMLOOP	Loops clip a given number of times.	no	any number	(none)	page 132
REGION	Ties clip to SMIL region.	no	SMIL region	(none)	page 133
SHUFFLE	Randomizes playback.	no	true|false	false	page 132
SRC	Specifies source clip.	**yes**	URL	(none)	page 119
WIDTH	Sets window or control width.	**yes**	percentage or pixels	(none)	page 120

INSERTING ADS IN A PRESENTATION



RealSystem's Advertising Application and SMIL make it easy to insert advertisements into your streaming presentation. You can include GIF ad banners, for example, or have a video ad precede requested content. This chapter explains how to place ads in SMIL presentations. Before you start, you should understand SMIL as described in Chapter 7 beginning on page 75.

How Ad Serving Works

To use RealSystem's Advertising Application, you create a SMIL file that lays out requested clips and ad clips. Instead of listing ad clip URLs in SMIL tags such as `<img/>` or `<video/>`, you write `<RealAdInsert/>` tags that cause RealServer to insert ad URLs automatically. RealServer typically gets these ad URLs from a third-party ad server. RealSystem can deliver ads in any format played by RealPlayer, including GIF, animated GIF, JPEG, RealAudio, RealVideo, and RealFlash.

The URL used to request the SMIL file, not the `<RealAdInsert/>` tag, determines what type of ad appears in place of each `<RealAdInsert/>` tag. For example, your Web page might contain a hyperlink to a SMIL file that looks like this:

```
<a href="http://realserver.company.com/ramgen/adtag/banner/start.smil">...</a>
```

Here, the URL component `/adtag/banner/` indicates that the SMIL file contains one or more `<RealAdInsert/>` tags that RealServer replaces with ad URLs. The RealServer administrator might define `/adtag/banner/` to mean that each `<RealAdInsert/>` tag is replaced by a single, 486-pixel by 60-pixel banner ad. A URL component such as `/adtag/rotating_banner/` might indicate banner ads that rotate every 30 seconds. Another component such as `/adtag/lead_in/` might be for RealVideo ads that precede the requested content.

To use RealSystem's Advertising Application, you need to work closely with your RealServer administrator. The administrator can tell you the sizes and file formats of ads available for streaming, the URLs you'll need to use in your Web pages, and other important information. RealServer can also generate SMIL files containing ads, or insert ads in existing SMIL presentations. This lets you serve ads for existing content without having to write or modify SMIL files.

Tip

Check with your RealServer administrator on the availability of the SMIL generation feature for ad streaming. This feature can save development time and effort when you want to include ads with existing clips or SMIL files.

Displaying Banner Ads

RealServer can stream a single banner ad or a series of rotating banner ads within a SMIL presentation. Each ad includes a hyperlink that, when clicked, opens the URL in the viewer's Web browser. The RealServer administrator defines the banner ad size, and determines how much bandwidth each ad uses. For rotating banner ads, the administrator determines how frequently RealPlayer displays a new ad.

Note

Before creating SMIL files with banner ads, check with the administrator on banner ad size, bandwidth, type (single ad or rotating ads), and rotation frequency.

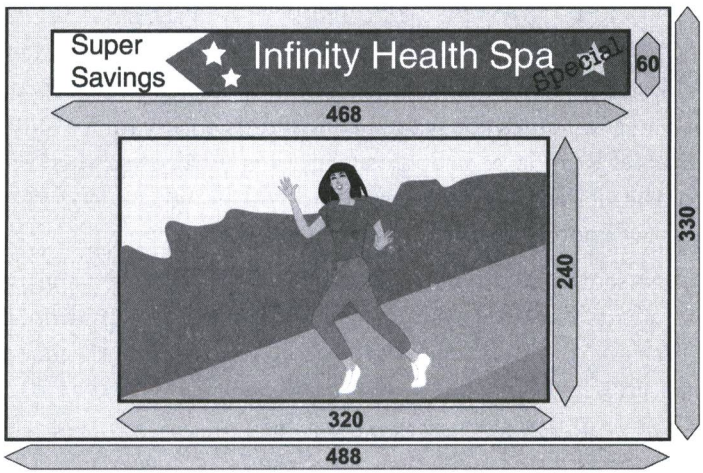
Laying Out the Banner Region

To integrate banner ads with your presentation, you create a SMIL region in which the ad image or rotating image series displays. The following SMIL file creates an ad banner region above a video region. The video region is 320 pixels wide by 240 pixels high. The ad banner region is 468 pixels wide by 60 pixels high, a common size for ad banners. In the SMIL body section, a `<par>` group plays the requested video and banner ads in their respective regions:

```
<smil>
  <head>
    <!--presentation with video clip and ad banner-->
    <layout>
      <root-layout width="488" height="330" background-color="black"/>
      <region id="ad_banner" top="10" left="10" width="468" height="60"/>
      <region id="videoregion" top="80" left="84" width="320" height="240"/>
    </layout>
  </head>
  <body>
    <par>
      <RealAdInsert region="ad_banner" dur="9:00.0" fill="freeze"/>
      <video src="rtsp://realserver.company.com/videos/newsvid.rm"
        region="videoregion"/>
    </par>
  </body>
</smil>
```

The following figure illustrates this layout of a banner ad region above a requested video clip.

Ad Banner Layout



Additional Information

"Laying Out Multiple Clips" on page 95 explains how to set up SMIL regions.

Setting Banner Ad Properties

In the SMIL body, instead of an <img/> tag link to an ad image file, you use a <RealAdInsert/> tag that triggers automatic ad insertion. In the SMIL example above, the ad insertion tag looks like this:

```
<RealAdInsert region="ad_banner" dur="9min" fill="freeze"/>
```

When RealPlayer requests a SMIL file with an ad insertion tag, RealServer replaces the <RealAdInsert/> tag with a SMIL <img/> tag that includes a src attribute pointing to a banner ad or, with ad rotation, a series of banner ads. The <RealAdInsert/> tag can also contain SMIL attributes such as region="ad_banner" and fill="freeze". RealServer simply includes these SMIL attributes in the replacement <img/> tag. A duration time is needed only when using rotating banner ads.

Specifying Durations for Rotating Banner Ads

To use rotating banner ads with a live broadcast, do not set a duration time in the <RealAdInsert/> tag. RealServer then sends new banner ads to RealPlayer as long as it plays the broadcast. For prerecorded content, however, set a duration approximately equal to the content length. If the requested content is nine minutes long, for example, you would set the ad duration as dur="9min".

You specify an ad rotation duration when serving prerecorded content so that RealSystem can calculate the presentation length. If you set no duration, RealServer assumes you want a continuous ad stream for a live broadcast, and it disables RealPlayer's clip position slider. RealServer then sends new ads until the viewer halts the presentation. Although the presentation will play OK, it will not fast-forward or rewind.

You can set the ad duration shorter than the requested content if you do not want ads to display for the entire length of the presentation. In this case, a fill="remove" or fill="freeze" attribute included in the <RealAdInsert/> tag determines if the last ad banner disappears or remains on the screen.

Do not set the ad duration significantly higher than the length of the content, however. RealSystem considers the ad duration when calculating the presentation's total running time. If you set an ad duration of 20 minutes with a 10-minute video, for example, RealPlayer indicates in its status bar that the presentation lasts 20 minutes. If a viewer moves the RealPlayer clip position slider forward to the 15-minute mark, for instance, the video will stop. The viewer may conclude that the presentation is flawed.

Additional Information

See "Specifying Timing" on page 84 for more on SMIL timing values.

Interleaving Rich Media Ads with Clips

Rich media advertising is quickly gaining popularity, with many ad serving companies offering rich media ads in formats such as RealVideo or RealFlash. With the RealSystem Advertising Application, you can play rich media ads before, between, or within requested clips. You simply create a SMIL file that defines when and how ads appear in relation to requested clips. Instead of supplying URLs to ad clips, you write a `<RealAdInsert/>` tag that causes RealServer to insert the rich media ad automatically.

Note

The URL for the SMIL file determines the type of rich media ad inserted in place of `<RealAdInsert/>`. Your RealServer administrator sets up these request URLs, and can give you information about rich media ad sizes, formats, and bandwidths.

Tip

Always account for the higher bandwidth requirements of rich media ads. Ensure that your target audience has sufficient bandwidth to receive the ad as well as the requested clips. For more on bandwidth, see Chapter 3 beginning on page 27.

Creating a SMIL File for a Rich Media Ad

To use rich media advertising, create a SMIL file that organizes all clips, whether ad clips or requested content. For a rich media ad, use a `<RealAdInsert/>` tag in place of a clip source tag such as `<video/>`. Rich media ads typically play before or between requested clips. Their number, placement, and timing depend entirely on how you construct the SMIL file.

SMIL File with No Regions

The following SMIL file has no layout. It simply plays two video clips in sequence. The first clip is the rich media ad:

```
<smil>
 <body>
  <seq>
   <RealAdInsert/>
   <video src="rtsp://realserver.company.com/video/video.rm"/>
  </seq>
 </body>
</smil>
```

SMIL File with No Regions

In the preceding example, the lack of SMIL regions causes the RealPlayer display window to resize automatically for each clip. To prevent resizing, or to lay out clips that play together, define SMIL regions for the clips. Here's an example of one video region defined for two clips played in sequence:

```
<smil>
 <head>
  <layout>
   <root-layout width="240" height="180"/>
   <region id="videoregion" width="240" height="180"/>
  </layout>
 </head>
 <body>
  <seq>
   <RealAdInsert region="videoregion"/>
   <video src="rtsp://realserver.company.com/video/video.rm"
      region="videoregion"/>
  </seq>
 </body>
</smil>
```

The `<RealAdInsert/>` tag above includes the SMIL `region` attribute to associate the ad with the correct SMIL region. You can include other SMIL attributes, such as the `dur` or `fill` attribute, in a `<RealAdInsert/>` tag as well.

Additional Information

"Laying Out Multiple Clips" on page 95 explains how to set up SMIL regions.

Disabling the RealPlayer Playlist

When a rich media ad and a requested clip play in sequence, the viewer can skip the ad clip by choosing the requested clip in the RealPlayer playlist. To