



## **REALPRODUCER COMMAND LINE AND JOB FILE REFERENCE**

RealProducer 13.1

Revision Date: 18 December 2009

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

<http://www.real.com>  
<http://www.realn networks.com>

Copyright ©2004-2009 RealNetworks, Inc. All rights reserved.

Information in this document is subject to change without notice. All rights reserved. No part of this book may be reproduced, in any form or by any means, without permission in writing from RealNetworks, Inc.

Helix, the Helix logo, the Real "bubble" (logo), RealProducer, Helix Producer, RealSystem Server, Helix Universal Server, RealAudio, RealVideo, RealMedia, RealPlayer, and RealOne Player are all trademarks or registered trademarks of RealNetworks, Inc.

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

-----  
Copyright (c) 1995-2009 RealNetworks, Inc. This product may incorporate one or more of the following: U.S. Patent # 5,917,835; U.S. Patent # 5,854,858; U.S. Patent # 5,917,954; U.S. Patent # 6,597,961; U.S. Patent #6,314,466. Other U.S. patents pending. All rights reserved.

RealNetworks Lossless audio codec Copyright (c) 1999-2003 RealNetworks, Inc. All rights reserved.

RealNetworks RealAudio Multichannel Codec Copyright (c) 2003 RealNetworks, Inc. All rights reserved.

ACELP(r).net codec by VoiceAge Corporation Copyright(c) 2000-2002. All rights reserved.

AAC and aacPlus implementation developed by Coding Technologies. All rights reserved.

The Ogg Multimedia Framework and the Vorbis audio compression tools have been provided by the Xiph.org Foundation.

RealNetworks RealVideo 8 video codec Copyright (c) 1995-2004 RealNetworks, Inc. Portions Copyright (c) 1999-2000 Intel Corporation. All rights reserved.

RealNetworks RealVideo 9 video codec Copyright (c) 1995-2004 RealNetworks, Inc. Portions Copyright (c) 1999-2003 Intel Corporation. All rights reserved.

RealNetworks RealVideo 10 video codec Copyright (c) 1995-2004 RealNetworks, Inc. Portions Copyright (c) 1999-2003 Intel Corporation. All rights reserved.

STLport (c) 1999, 2000 Boris Fomitchev. Copyright (c) 1994 Hewlett-Packard Company. Copyright (c) 1996,97 Silicon Graphics Computer Systems, Inc. Copyright (c) 1997 Moscow Center for SPARC Technology.

Helix, RealAudio, RealNetworks, RealSystem, RealVideo, and SureStream are trademarks or registered trademarks of RealNetworks, Inc. All other companies or products listed herein are trademarks or registered trademarks of their respective owners. All rights reserved.

# CONTENTS

INTRODUCTION	1
How This Guide Is Organized .....	1
Conventions Used in this Guide.....	2
 PART I: FILE SYNTAX	
1    JOB FILE	7
Job File Overview .....	7
Job File Structure .....	8
Job Section.....	10
Job File Example.....	10
Job Attributes.....	10
Metadata .....	13
Metadata Examples .....	13
Metadata Attributes .....	14
Metadata Values for RealMedia .....	16
Inputs Section .....	16
Single Input.....	16
Multiple Inputs .....	17
File Input .....	17
File Input Example.....	18
File Input Attributes.....	18
Capture Input .....	20
Live Capture Examples.....	20
Capture Input Attributes.....	21
Live Input Devices and Ports.....	24
Outputs Section .....	25
Output Syntax.....	26
Output .....	28
Output Examples .....	28
Output Attributes.....	28
File Destination .....	30
File Destination Example.....	31
File Destination Attributes .....	31

2	PREFILTERS	35
	Prefilter Syntax .....	35
	Available Prefilters .....	35
	Prefilter Order for Video Input.....	36
	Video Input Cropping.....	37
	Input Cropping Example.....	37
	Input Cropping Filter Attributes .....	37
	De-Interlace and Inverse-Telecine .....	39
	De-Interlace and Inverse-Telecine Examples .....	39
	De-Interlace and Inverse-Telecine Filter Attributes .....	40
	Video Noise Reduction .....	41
	Noise Reduction Example .....	41
	Noise Reduction Filter Attributes .....	41
	Black Level .....	42
	Black-Level Example .....	43
	Black-Level Filter Attributes.....	43
	Video Resizing.....	43
	Video Resizing Examples .....	44
	Video Resizing Filter Attributes.....	45
	Width and Height Values for Resizing.....	46
	Audio Gain .....	47
	Audio Gain Example.....	47
	Audio Gain Prefilter Attributes .....	47
	Audio Delay Compensation .....	48
	Audio Delay Compensation Example.....	49
	Audio Delay Compensation Filter Attributes .....	49
3	AUDIENCE FILE	51
	Audience Section.....	51
	Audiences in Job Files .....	51
	Audience Templates .....	52
	Audience Element.....	53
	Audience Example .....	53
	Audience Properties.....	54
	Audio Stream Properties.....	56
	Audio Stream Examples .....	56
	Audio Stream Attributes .....	56
	Video Stream Properties.....	59
	Video Stream Example.....	59
	Video Stream Attributes.....	60
	Rate Control Method .....	65
	Packetizers Section .....	67

	Packetizer Element.....	67
	Packetizer Example.....	67
	Packetizer Attributes.....	68
	Maximum Packet Size and Duration.....	69
	Maximum Superblock Size.....	70
	Latency Mode and Maximum Packet Duration.....	71
4	SERVER FILE.....	73
	Server Destinations.....	73
	Server Destinations in Job Files.....	73
	Server Destination Template Files.....	74
	Server Element.....	74
	Push Server Destinations.....	75
	Push Broadcast Types.....	75
	Push Server Attribute Summary.....	75
	Push Server Properties.....	76
	Push Server Examples.....	82
	Pull Server Destination.....	83
	Pull Server Attribute Summary.....	83
	Pull Server Properties.....	84
	Pull Server Example.....	86
PART II: COMMAND LINE		
5	COMMAND-LINE BASICS.....	91
	Command-Line Application.....	91
	Running the Application.....	91
	Stopping the Application.....	92
	Working Directory and Relative Paths.....	92
	Output and Temporary Directories.....	92
	Command-Line Encoding.....	93
	Basic Command Options.....	93
	Syntax Notes.....	93
	Syntax Changes.....	94
	Job File Processing.....	96
	Compatibility with Earlier Job Files.....	96
	Windows Signaling.....	96
	Signal Producer Utility on Windows.....	97
	Return Values on Windows.....	97
6	INPUT OPTIONS.....	99
	Job Files.....	99
	Job File Options.....	99

- Job File Examples ..... 102
- Inputs..... 103
  - Input Options ..... 104
  - File Input and Output Examples ..... 108
  - Live Capture Examples..... 109
- Metadata..... 110
  - Metadata Options ..... 110
  - Metadata Examples..... 112
- Prefilters ..... 113
  - Prefilter Options..... 113
  - Prefilter Examples..... 116
- 7 OUTPUT OPTIONS ..... 119
  - Outputs..... 119
    - Output Scopes..... 119
    - Output Options ..... 120
  - File Destinations ..... 121
  - Server Destinations..... 124
    - Server Destination Options ..... 124
    - Additional Server Parameters ..... 129
  - Audiences ..... 130
    - Audience Options ..... 130
    - Audience Examples..... 132
- 8 GENERAL OPTIONS ..... 135
  - Logging Options ..... 135
  - Help Options ..... 139
- PART III: APPENDIXES
- A AUDIENCE TEMPLATES ..... 143
  - Predefined Audience Templates..... 143
    - Templates Included with RealProducer ..... 143
    - Customized Templates ..... 144
  - 16k Voice..... 144
  - 28k Video ..... 145
  - 32k Music..... 146
  - 56k Video ..... 146
  - 64k Music..... 147
  - 65k Video ..... 148
  - 80k Video ..... 148
  - 96k Music..... 149
  - 100k Video ..... 150

150k Video.....	150
300k Video.....	151
750k Video.....	152
1.5M Video .....	153
90% VBR Quality Video .....	153
<i>B</i> PREFERENCES .....	155
Preferences File.....	155
File Path Preferences .....	155
File Path Examples.....	155
File Path Attributes.....	155
Output Preferences.....	157
Output Settings Example .....	157
Output Attribute Prefixes .....	157
Output Attributes.....	158
Log File Preferences .....	160
File Logging Example .....	160
File Logging Attributes .....	161
Screen Logging Preferences.....	163
Screen Logging Example.....	163
Screen Logging Attributes .....	164
Video Signal Detection Preferences .....	165
Video Conditions Detected .....	165
Video Signal Detection Attributes.....	166
Audio Signal Detection Preferences.....	168
Audio Conditions Detected.....	169
Audio Signal Detection Attributes .....	169
INDEX .....	173



# INTRODUCTION

This guide explains how to use the command-line application of RealProducer 13.1. It also provides a reference for the syntax used with the RealProducer 13.1 job, audience, server, and preference files.

**For More Information:** This guide supplements *RealProducer User's Guide*. Refer to that manual for general information about encoding topics, as well as instructions for using the graphical application of RealProducer 13.1.

## How This Guide Is Organized

The guide contains the following chapters.

### Chapter 1: Job File

Job files can define the encoding settings used by the graphical application and the command-line application. This chapter describes the job file syntax.

### Chapter 2: Prefilters

Prefilters modify the input stream before it is encoded. This chapter covers the prefilter syntax of the job file.

### Chapter 3: Audience File

This chapter explains the syntax of audience files, which define how clips and broadcasts are encoded.

### Chapter 4: Server File

Server files define the server destinations used to create live broadcasts. This chapter covers the server file syntax.

### Chapter 5: Command-Line Basics

Refer to this chapter for basic information about encoding clips or broadcasts using the RealProducer command-line application.

### Chapter 6: Input Options

This chapter explains the command-line options for selecting input and applying prefilters.

### Chapter 7: Output Options

The command-line options explained in this chapter control outputs and destinations.

### Chapter 8: General Options

These command-line options affect logging and the display of online help.

### Appendix A: Audience Templates

Audience files set encoding properties such as the streaming bit rate. This appendix describes the audience templates included with RealProducer.

### Appendix B: Preferences

This appendix explains how to edit the RealProducer preference file to set overall encoding preferences.

## Conventions Used in this Guide

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

Notational Conventions	
Convention	Meaning
<b>emphasis</b>	Bold text is used for in-line headings, user-interface elements, URLs, and e-mail addresses.
<i>terminology</i>	Italic text is used for technical terms being introduced, and to lend emphasis to generic English words or phrases.
syntax	This font is used for fragments or complete lines of programming syntax (markup).
<b>syntax emphasis</b>	Bold syntax character formatting is used for program names, and to emphasize specific syntax elements.
<i>variables</i>	Italic syntax character formatting denotes variables within fragments or complete lines of syntax.
[options]	Square brackets indicate values that you may or may not need to use. As a rule, when you use these optional values, you do not include the brackets themselves.

(Table Page 1 of 2)

**Notational Conventions**

Convention	Meaning
choice 1 choice 2	Vertical lines, or “pipes,” separate values that you can choose between.
...	Ellipses indicate nonessential information omitted from examples.

(Table Page 2 of 2)



## FILE SYNTAX

The chapters in this section explain the structure and syntax of the RealProducer job file, audience file, and server file.



## JOB FILE

This chapter describes the job file syntax. Using this information, you can edit a job file manually.

### Job File Overview

A job file records the full range of RealProducer settings used to encode a clip or broadcast. You can create job files, which use the file extension `.rpjf`, using any of the following methods:

- RealProducer graphical application

The RealProducer graphical application can open, edit, and save job files. It also provides error checking to help prevent incorrect attribute values.

**For More Information:** *RealProducer User's Guide* explains how to use the graphical application.

- RealProducer command-line application

When you encode a clip or broadcast using the command-line application, you can save the settings to a job file. You can also process an existing job file from the command line.

**For More Information:** See “Job Files” on page 99.

- text or XML editor

You can edit a job file using any text or XML editor. However, you should first be familiar with XML syntax conventions such as namespaces, elements, attributes, and values.

**Tip:** When creating a new job file, start with an existing job file that you have renamed. RealProducer provides predefined job files in its `samples/jobs` subdirectory.

**For More Information:** For background on XML, refer to <http://en.wikipedia.org/wiki/XML>, for example.

## Job File Structure

The following example illustrates the main elements within a job file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Job ...>
  <Metadata>
    ...overall stream information such as title, author, and copyright...
  </Metadata>
  <ParInputs>
    <Input ...>
      ...input media, either a file or live capture data...
      <Prefilters>
        <Prefilters>
          ...prefilters to apply to the input...
        </Prefilters>
      </Prefilters>
    </Input>
  </ParInputs>
  <ParOutputs>
    <Output>
      <Destinations>
        <Destination>
          ...output destination, either an encoded clip or a broadcast...
        </Destination>
      </Destinations>
      <Audiences>
        <Audience>
          ...output audience that defines the overall encoding parameters...
          <Streams>
            <Stream ...audio stream encoding parameters.../>
            <Stream ...video stream encoding parameter.../>
          </Streams>
        </Audience>
      </Audiences>
      <Metadata>
        ...stream information specific to the output...
      </Metadata>
    </Output>
  </ParOutputs>
</Job>
```

**Note:** The order of most elements within the <Job> section does not matter. For example, the <ParInputs> element can precede the <Metadata> element. The hierarchy does matter, however. The <ParInputs> element cannot fall within the <Metadata> element, for instance.

### Job File Main Elements

The following table summarizes the main elements of the job file.

Job File Main Elements			
Element	Function	Parent	Reference
Job	Main container element. Defines namespaces and elements that affect overall job processing. Must come first in the file.	none	page 10
Metadata	Sets metadata values. As a child of Job, defines values for all outputs. As a child of Output, defines values for a single output	Job Output	page 13
ParInputs	Container for one or two inputs.	Job	page 16
Input	Container for a single input.	ParInputs	page 16
Prefilters	Container for input prefilters.	Input	page 35
Prefilter	Applies a specific filter to the input.	Prefilters	page 35
ParOutputs	Container for one or more outputs.	Job	page 25
Output	Container for a single output.	ParOutputs	page 25
Destinations	Container for one or more encoding destinations.	Output	page 26
Destination	Defines an encoding destination, either a file or a broadcast server.	Destinations	page 26
Audiences	Container for one or more audiences.	Output	page 51
Audience	Defines properties for a single audience.	Audiences	page 53
Streams	Container for audio and video streams.	Audience	page 51
Stream	Sets encoding parameters for an audio or video stream.	Streams	page 56 page 59

### Job File 3.0 Format

RealProducer 13.1 uses the job file 3.0 format. The older, version 2.0 job file format used with RealProducer 10 and 11, as well as earlier versions of Helix Mobile Producer, is **not** forward-compatible with RealProducer 13.1. However,

RealProducer 13.1 can read 2.0 job files and automatically update those files to the new 3.0 format.

**Note:** Once existing job files are updated to the 3.0 format, they are no longer compatible with earlier versions of RealProducer and Helix Mobile Producer.

**For More Information:** See “Job File Syntax Update” on page 102.

## Job Section

The <Job> and </Job> tags encapsulate all job information within a job file. The <Job> element defines the appropriate namespaces for the job. Additionally, it can define the following properties:

EnableTwoPass	Enables two-pass encoding for clips.
DisableLoadManagement	Shuts off broadcasting load management.
DisableWallclock	Disables codec wallclock timing.
LoadManagementJitterThreshold	Affects sensitivity to processing load variations.
LoadManagementLatencyMax	Defines point at which severe load management features are implemented.
LoadManagementLatencyThreshold	Determines when load management features are used.

## Job File Example

The following example shows the major structure of a job file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Job xmlns="http://ns.real.com/tools/job.3.0.0"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://ns.real.com/tools/job.3.0.0
      http://ns.real.com/tools/job.3.0.0.xsd"
      EnableTwoPass="true" DisableLoadManagement="false">
  ...all job file data...
</Job>
```

## Job Attributes

In addition to the required namespaces, the <Job> element can include the following attributes.

### EnableTwoPass

Allows two-pass encoding. If enabled, the first pass gathers data about the entire input file. The second pass encodes the clip.

value:	true false
default:	true (recommended)
required:	no

**Tip:** This setting is ignored for live broadcasts.

### DisableLoadManagement

Disables RealProducer's ability to reduce encoding complexity during live broadcasts. Load management enables RealProducer to keep up with the input stream if processor utilization is too high.

value:	true false
default:	true
required:	no

**Note:** Contact a RealNetworks support representative before changing this value to false.

### DisableWallclock

Disables wallclock timing in the codec.

value:	true false
default:	true (recommended)
required:	no

**Note:** Change this value only under the direction of RealNetworks support personnel.

### LoadManagementJitterThreshold

Defines a duration in seconds for which increases in encoding latency are ignored during a live broadcast. This value is ignored if `DisableLoadManagement` is set to true.

value:	s[.xyz] to 60
default:	1.5
required:	no

### What is the jitter threshold?

The jitter threshold affects how quickly RealProducer reacts to processing load during a live broadcast. It prevents RealProducer from reducing the encoding complexity during temporary spikes in the processing load. These spikes typically occur because of operating system conditions that pass quickly.

### When should I change the jitter threshold?

Observe the RealProducer encoding statistics while broadcasting:

- If RealProducer frequently reduces the encoding complexity level for a few seconds before resuming the normal complexity level, *increase* the jitter threshold. This makes RealProducer less sensitive to temporary bottlenecks caused by the operating system.
- If RealProducer frequently and suddenly decreases the encoding complexity by a large amount, *decrease* the jitter threshold value. This makes RealProducer faster to react to bottlenecks.

### LoadManagementLatencyMax

Sets the maximum amount of encoding latency allowed during a live broadcast. This value is ignored if `DisableLoadManagement` is set to true.

value:	s[.xyz] to 60
default:	1.5
required:	no

### LoadManagementLatencyThreshold

The maximum amount of encoding latency allowed during a live broadcast before RealProducer begins to reduce the encoding complexity. This value is ignored if `DisableLoadManagement` is set to true.

value:	s[.xyz] to 60
default:	0.5
required:	no

### What is encoding latency?

During a live broadcast, encoding latency is the time between data capture and data encoding:

- If RealProducer detects that the encoding latency is greater than the specified threshold value, it begins to reduce the encoding complexity to give the processor time to catch up with that capture input stream.
- If the latency becomes greater than the specified maximum value, RealProducer takes significant action to reduce the processor load. This typically involves dropping frames from video streams.

**Tip:** Higher values for the threshold and maximum attributes allow RealProducer to keep the encoding quality high during CPU usage spikes. However, if encoding time lags too far behind capture time, broadcast servers and media players that buffer little stream data (such as one second), may need to pause playback until new stream data arrives.

## Metadata

The optional metadata section defines informational values that are encoded into the stream. These values are provided for the benefit of the viewer, and do not affect the encoded content.

### Metadata Examples

Each metadata value falls between `<Entry>` and `</Entry>` tags within the `<Metadata>` section:

```
<Metadata>
  <Entry Name="name" Type="type">Value</Entry>
  ...additional metadata entries...
</Metadata>
```

### Same Metadata Used for All Outputs

If all metadata information is defined only within the main `<Metadata>` element, the values are used for all outputs:

```
<Job...>
  <Metadata>
    <Entry Name="Copyright">(C) 2009</Entry>
    <Entry Name="Keywords">sports football</Entry>
    <Entry Name="Description">Highlights of the 2008 football season.</Entry>
  </Metadata>
  ...all other job file information...
</Job>
```

## Different Metadata Used for Different Outputs

You can define separate metadata values for each output. In this case, metadata specific to the output overrides any corresponding entry in the main <Metadata> element.

The following example sets basic metadata such as title, author, and copyright for both outputs, but uses separate descriptions for the broadcast and clip outputs:

```
<Job...>
  <Metadata>
    <Entry Name="Title">2008 Football Highlights</Entry>
    ...other metadata values such as author and copyright...
  </Metadata>
  ...input information...
  <ParOutputs>
    <Output>
      ...output information for the live broadcast...
      <Metadata>
        <Entry Name="Description">A live, year-end review
          of the highlights of the 2008 football season.</Entry>
      </Metadata>
    </Output>
    <Output>
      ...output information for the archive clip...
      <Metadata>
        <Entry Name="Description">Archived highlights of
          the 2008 fall football season.</Entry>
      </Metadata>
    </Output>
  </ParOutputs>
  ...additional job file information...
</Job>
```

**For More Information:** See “Outputs Section” on page 25.

## Metadata Attributes

The following are the attributes and value of the metadata <Entry> element.

**Name**

---

Defines the type of metadata.

value:	string up to 255 bytes
default:	none
required:	yes

**Note:** A media player must recognize the name to handle the metadata value properly.

**Type**

---

Indicates the type of value so that the media player can handle the data correctly.

value:	boolean double string unsignedint
default:	string
required:	no

**Value**

---

Supplies the value for the metadata field.

value:	string up to 64 Kilobytes
default:	none
required:	no

**Tip:** The value can be a binary object stored in a CDATA element.

## Metadata Values for RealMedia

RealMedia outputs can encode the following metadata values.

**Metadata Properties for RealMedia Outputs**

Name	Function
Author	Indicates the clip author.
Content Rating	Assigns a number that defines an appropriate age range for viewers: 0–No Rating (this is the default) 1–All Ages 2–Older Children 3–Younger Teens 4–Older Teens (15 and up) 5–Adult Supervision Recommended 6–Adults Only
Copyright	Provides the clip copyright.
Description	Gives a description that is more informative than the title.
Keywords	Adds keywords for search engines that can read RealMedia clips. Separate keywords with spaces.
Title	Adds a clip title.

## Inputs Section

The inputs section of the job file defines the audio or video input for the job. Inputs can be either digitized files or output from an audio or video capture card. The `<ParInputs>` and `</ParInputs>` tags encapsulate the inputs section. Within this list, you can define one or two inputs using `<Input>` elements.

**For More Information:** Refer to *RealProducer User's Guide* for information about acceptable input file formats.

### Single Input

To define inputs, you add `<Input>` and `</Input>` tags within the `<ParInputs>` element. The following specifies a single input:

```
<ParInputs>
  <Input...>
    ...single input defined here...
  </Input>
</Inputs>
```

## Multiple Inputs

You can also define multiple inputs. This allows you to encode audio from one file and video from a different file. Or, if you have separate audio and video capture cards, encode the captured output from each:

```
<ParInputs>
  <Input...>
    ...audio input defined here...
  </Input>
  <Input...>
    ...video input defined here...
  </Input>
</ParInputs>
```

**Note:** If your video capture card captures both video and audio, define a single input for it.

**Tip:** When using parallel inputs, the audio delay compensation prefilter can synchronize the audio and video inputs. See “Audio Delay Compensation” on page 48.

**Warning!** When a job includes parallel inputs, you **must** encode the job through the command-line application. The graphical application supports only one file or capture input.

## File Input

To encode from a digitized input file, you add an `<Input>` element to the `<ParInputs>` list. The following are the attributes available for use for a file input.

EndPosition	Stops the encoding at a certain point in the file.
Filename	Supplies the path and name of the input file.
Name	Assigns a name to the source.
PluginName	Provides the name of the plug-in used to read the input file.
StartPosition	Starts the encoding at a certain point in the file.
xsi:type	Indicates a file input by specifying the value <code>AvFileInput</code> .

**For More Information:** See “Inputs Section” on page 16 for details about the syntax of an inputs list.

## File Input Example

The following example defines a single input from a digitized file:

```
<ParInputs>
  <Input xsi:type="AVFileInput" Filename ="C:\media\videos\video1.avi" />
</ParInputs>
```

## File Input Attributes

The <Input> element for file encoding can include the following attributes.

### EndPosition

Sets the timing mark within the file at which encoding automatically ends.

value:	[d:][h:][m:]s[.xyz]
default:	none
required:	no

**For More Information:** For examples of time values, refer to the section “Timing Values” on page 22.

### Filename

Indicates the path and name of the input file, using the file name and path conventions standard for your operating system.

value:	file name and path (wildcards <b>not</b> allowed)
default:	none
required:	optional if using the graphical interface; required if using the command line

#### When to Include the File Name

Whether you need to specify the Filename depends on how you encode:

- command line

When encoding using the command line, include the path and file name in the job file.

- graphical application

When using the graphical application, you can specify the name in the job file. Or, leave the file name blank and supply the name using the graphical application.

### Relative and Absolute Paths

You can specify a relative or absolute path to a file. Relative paths are relative to the current working directory:

- command line

The command-line application's working directory is the directory where the application is run. If the job file's paths are relative to the job file directory, run the application from that directory.

- graphical application

If you encode a job using the graphical application, the working directory automatically becomes the directory that holds the job file.

### Name

---

Assigns a name to the source, such as source1, source2, and so on.

value:	string
default:	none
required:	no

**Note:** Names are recommended when using parallel inputs.

### PluginName

---

Provides the name of the file system plug-in used to read the input file.

value:	plug-in name
default:	rn-avreader-all
required:	no

#### When to Specify a Plug-In Name

If you omit the PluginName value property, RealProducer scans available plug-ins and uses the first one it finds that can read the input format. In most cases, this is desirable. If you are using a customized plug-in, however, specify the PluginName value.

### StartPosition

---

Sets the timing mark within the file at which encoding starts.

value:	[d:][h:][m:]s[.xyz]
--------	---------------------

default:	0
required:	no

**For More Information:** For examples, see “Timing Values” on page 22.

xsi:type

Indicates that the input is a file.

value:	AVFileInput
default:	none
required:	yes

## Capture Input

To encode output from a capture card, you add an <Input> element to the <ParInputs> list. The following are attributes used for live capture input.

AudioDeviceID	Sets the audio device used for recording.
AudioDevicePort	Specifies the audio port used for recording.
Duration	Indicates length of time to capture from the device.
Name	Assigns a name to the source.
PluginName	Provides the name of the plug-in used to read the input.
VideoDeviceID	Sets the video device used for the input.
VideoDevicePort	Specifies the video port used for the input.
VideoFrameHeight	Sets a video capture height in pixels.
VideoFramerate	Sets the frame rate at which to capture input.
VideoFrameWidth	Sets a video capture width in pixels.
xsi:type	Indicates a file input by specifying the value CaptureInput.

## Live Capture Examples

The following are examples of capturing live input.

### Live Capture for Limited Time

This example captures audio and video input for 15 minutes once encoding begins.

```
<ParInputs>
  <Input xsi:type="CaptureInput" AudioDeviceID="Sound Blaster *1"
    AudioDevicePort="Mic*" VideoDeviceID="Osprey Capture Card 1"
    VideoDevicePort="S-Video" VideoFramerate="15.00"
    Duration="15:00.000" />
</ParInputs>
```

### Parallel Inputs

Using parallel inputs, you can encode audio and video from separate sources. The following example captures the video track from a digitized file. It combines this stream with five minutes of audio from live audio input.

```
<ParInputs>
  <Input xsi:type="AVFileInput" Filename="c:\media\promo.avi" />
  <Input xsi:type="CaptureInput" AudioDeviceID="Sound Blaster *1"
    AudioDevicePort="Mic*" PluginName="rn-capture-av" Duration="5:00.000" />
</ParInputs>
```

**For More Information:** See “File Input” on page 17.

## Capture Input Attributes

The <Input> element for live capture can include the following attributes.

### AudioDeviceID

---

Sets the audio device used for recording.

value:	integer string
default:	none
required:	yes, to capture audio from an audio or video capture card

**For More Information:** See “Live Input Devices and Ports” on page 24.

### AudioDevicePort

---

Specifies the audio port used for recording. You must also provide the device ID.

value:	integer string
default:	none
required:	no

**For More Information:** See “Live Input Devices and Ports” on page 24

### Duration

---

Sets the length of time to capture from the device.

value:	[d:][h:][m:]s[.xyz] infinite
default:	infinite (encode until RealProducer stops the job)
required:	no

### Timing Values

Only the seconds value is required. If a value is omitted, it is assumed to be zero. You must specify intermediate values. To indicate hours, for example, you must include the minutes and seconds field. Here are some sample values:

30	30 seconds
45.5	45-1/2 seconds
5:35	5 minutes, 35 seconds
1:0:0	1 hour
1:22:30:0	1 day, 22 hours, 30 minutes

**Tip:** You can specify down to a fraction of a second, but the timing is accurate only to the packet size of the stream. For audio this is about 200 milliseconds. For video, the size varies with the bit rate.

### Name

---

Assigns a name to the source, such as source1, source2, and so on.

value:	string
default:	none
required:	no (recommended when using multiple inputs)

### PluginName

---

Provides the name of the plug-in used to read the input.

value:	plug-in name
default:	rn-capture-av
required:	no

### When to Specify a Plug-In Name

If you omit the `PluginName` value property, `RealProducer` scans available plug-ins and uses the first one it finds that can read the input format. In most cases, this is desirable. If you are using a customized plug-in, however, specify the `PluginName` value.

### VideoDeviceID

---

Sets the video device used for recording.

value:	integer string
default:	none
required:	yes, to capture video from a video capture card

**Note:** If you use a single video card to capture both video and audio, you must specify both the `AudioDeviceID` and `VideoDeviceID` parameters.

**For More Information:** See “Live Input Devices and Ports” on page 24.

### VideoDevicePort

---

Specifies the video port used for recording. You must also provide the device ID.

value:	integer string
default:	none
required:	no

**For More Information:** See “Live Input Devices and Ports” on page 24.

### VideoFrameHeight

---

Sets an optional video height in pixels. Use any positive integer divisible by 4.

value:	integer up to 2048
default:	0 (use the capture device’s default size)
required:	no

**Tip:** In an audience’s video stream settings, you can specify a separate height for each output. See “OutputHeight” on page 63.

### VideoFramerate

---

Sets the frame rate (fps) at which to capture input, such as 15.00.

value:	double-precision number from 0 to 60.
default:	capture card default value
required:	no

### Input and Output Frame Rates

Video is usually captured at 15 fps or 30 fps. Frame rates for encoded outputs vary. The video stream properties of each audience set a maximum output frame rate. The output frame rate can never exceed the input frame rate.

**Tip:** To utilize processing power efficiently, set the input frame rate to the same value as the highest output frame rate. For output frame rates, see “MaxFramerate” on page 62.

### VideoFrameWidth

---

Sets a video width in pixels. Use any positive integer divisible by 4.

value:	integer up to 2048
default:	0 (use the capture device’s default size)
required:	no

**Tip:** In an audience’s video stream settings, you can specify a separate width for each output. See “OutputWidth” on page 63.

### xsi:type

---

Indicates that the input is live.

value:	CaptureInput
default:	none
required:	yes

## Live Input Devices and Ports

When capturing live input, the AudioDeviceID and VideoDeviceID parameters select the capture card or cards used for the input. The AudioDevicePort and VideoDevicePort parameters set the ports used for audio and video capture, respectively. You can use several types of values.

**Tip:** The command-line application's can display device information. See “-pd (print device information)” on page 139.

### Integer Value

An integer of value 0 or higher identifies a specific device or port number. A value of 0 always selects the first device or port. For example:

```
AudioDeviceID="1"
```

**Tip:** Using numbers for the device IDs maximizes the portability of the job file so that you can transfer it to another machine.

### String Value

A string value matches an existing device or port name:

```
AudioDeviceID="Sound Blaster Live!"
```

```
AudioDeviceID="Line In"
```

```
VideoDeviceID="Osprey Capture Card 1"
```

```
VideoDevicePort="S-Video"
```

### String Value with Wildcard

A string value can include a wildcard (\*) to match an existing device or port name, such as:

```
AudioDeviceID="Sound Blaster *"
```

```
AudioDeviceID="Mic*"
```

```
VideoDeviceID="Osprey Capture Card *"
```

```
VideoDeviceID="Composite*"
```

### UNIX Device Name

A string can match the UNIX device name, such as the following:

```
AudioDeviceID="/dev/audio"
```

```
VideoDeviceID="/dev/video"
```

## Outputs Section

The outputs section of a job file defines one or more outputs for the job. If you want to encode a single clip, for example, you define a single output.

Reasons for defining multiple outputs include the following:

- Encoding using different codecs.

Suppose that you want to encode RealMedia clips using both the RealVideo 8 and RealVideo 10 codecs. You would define a separate output for each codec.

- Encoding video at different screen sizes.

You would define separate outputs to encode a high-bandwidth clip at a large screen size, and a low-bandwidth clip at a smaller screen size.

## Output Syntax

The `<ParOutputs>` and `</ParOutputs>` tags encapsulate the outputs section. Within this list, you can define one or more outputs between `<Output>` and `</Output>` tags.

An `<Output>` element (see “Output” on page 28) is a container for additional elements that define the output’s destinations, encoding settings, and metadata:

```
<ParOutputs>
  <Output...>
    <Destinations>
      ...one or more output destinations, either a clip or a broadcast server...
    </Destinations>
    <Audiences>
      ...one or more audiences that define the encoding settings...
    </Audiences>
    <Metadata>
      ...optional clip information for the output...
    </Metadata>
  </Output>
  ...additional outputs...
</ParOutputs>
```

**Warning!** All outputs defined within the `<ParOutputs>` section are encoded simultaneously. When broadcasting, it is important that your RealProducer machine has sufficient processing power to encode all outputs in real-time.

### Destinations

The destinations section of an output tells RealProducer where to write the output. The `<Destinations>` and `</Destinations>` tags (note the plural) encapsulate the destinations section within an `<Output>` element. Within this

list, you can define one or more destinations between `<Destination>` and `</Destination>` tags (note the singular):

```
<ParOutputs>
  <Output...>
    <Destinations>
      <Destination>
        ...first output destination, such as a broadcast server...
      </Destination>
      <Destination>
        ...second output destination, such as a clip...
      </Destination>
    </Destinations>
    ...additional output data, such as audience settings...
  </Output>
  ...additional outputs...
</ParOutputs>
```

There are two possible types of destinations for each output:

- file

A file destination creates a clip. Typically, there is only clip destination for each output. See “File Destination” on page 30.

- broadcast server

A broadcast destination transmits the encoded data to a broadcast server. You define a separate broadcast destination for each server used. See “Server Destinations” on page 73.

**Tip:** Creating a server destination *and* a clip destination for the same output writes an archive clip for the live broadcast.

## Audiences

The `<Audiences>` element sets the encoding parameters used for the output. Settings include the video and audio codecs used, the video output size, and the streaming bit rates.

**For More Information:** See “Audience Section” on page 51.

## Metadata

The optional `<Metadata>` element defines stream information specific to the output. These values override any corresponding global values.

**For More Information:** See “Metadata” on page 13.

## Output

An <Output> element defines a single output. It accepts the following attributes:

LatencyMode	Sets a latency mode for a RealMedia broadcast.
MaxPacketDuration	Sets a maximum time before writing a packet.
MaxPacketInterleavingDuration	Sets RealAudio superblock size.
MaxPacketSize	Sets the maximum packet size.
Name	Defines a name for the output.
OutputType	Sets the output file type.

**For More Information:** For information about the syntax for defining outputs, see “Outputs Section” on page 25.

### Output Examples

The following example is for a low-latency broadcast in the RealMedia format:

```
<Output Name="Low-Latency Broadcast" OutputType="rm" LatencyMode="1">
  ...destinations, audiences, and metadata...
</Output>
```

### Output Attributes

You can include the following attributes in the <Output> element.

#### LatencyMode

Reduces latency for live broadcasts in the RealMedia format. Ignored for a clip destination.

value:	0 1 2
default:	0 (normal latency, highest error resiliency)
required:	no

**Tip:** You can also set the latency mode at the audience level (see “Audience Properties” on page 54). If both values are defined, the audience setting overrides the output setting.

#### Available Latency Modes

The LatencyMode property can decrease the end-to-end latency for live broadcasts in the RealMedia format:

- 0 – Normal system latency with high error resiliency.
- 1 – Lower system latency with lower resiliency for dropped packets.
- 2 – Lowest system latency with lowest resiliency for dropped packets.

**Warning!** Use a low latency setting only when delivering a live stream to Helix Server version 11 or later. Earlier versions of Helix Server cannot handle the stream properly.

**For More Information:** See “Latency Mode and Maximum Packet Duration” on page 71.

### MaxPacketDuration

Defines the maximum time that RealProducer can encode a RealAudio or RealVideo stream before writing a packet.

value:	s[.xyz] up to 60.000
default:	0 (RealProducer determines when to write packets)
required:	no

**For More Information:** See “Packetizers Section” on page 67.

### MaxPacketInterleavingDuration

Sets the superblock size for packetization of RealAudio streams.

value:	s[.xyz] up to 60.000
default:	codec default
required:	no

**For More Information:** See “Maximum Superblock Size” on page 70.

### MaxPacketSize

Sets the maximum size for a RealAudio or RealVideo packet, in bytes.

value:	integer up to 15000
default:	600, if bit rate is less than 128 Kbps 1000, if bit rate is 128 Kbps to 256 Kbps 1352, if bit rate is greater than 256 Kbps
required:	no

**For More Information:** See “Packetizers Section” on page 67.

## Name

---

Assigns a handle to the output.

value:	string
default:	"Output <i>n</i> " ( <i>n</i> starts at 1 and increments for each output)
required:	no (recommended if using the graphical application)

**Note:** This is used only by the graphical application to label the output.

## OutputType

---

Defines the output file type.

value:	ra – RealAudio output with a .ra file extension rm – RealAudio or RealVideo output with a .rm file extension rv – RealVideo output with a .rv file extension
default:	none
required:	yes

**Note:** For RealMedia, use the same OutputType value for either single-rate or multi-rate output.

## File Destination

A file destination writes the output to a digitized clip. To write output to a file, define a <Destination> element, which can specify the following attributes:

DestinationRollSize	Sets the size at which to create a new output clip.
DestinationRollTime	Sets the time at which to create a new output clip.
Filename	Indicates the path and name of the out file.
Name	Assigns a handle to the output.
PluginName	Provides the name of the plug-in that writes the file.
xsi:type	Indicates a file output using the value FileDestination.

**For More Information:** For information about the overall syntax for defining outputs and destinations, see “Outputs Section” on page 25.

**Note:** The parent <Output> element determines the overall output type. See “Output” on page 28.

## File Destination Example

The following example illustrates how to write an output to a digitized file:

```
<Destination xsi:type="FileDestination" Filename="C:\media\movie.rm"
/>
```

## File Destination Attributes

The following attributes affect file destinations.

### DestinationRollSize

Sets the size in Megabytes at which to write a new RealMedia output file.

value:	integer up to 4906
default:	0 (no rolling unless maximum size reached)
required:	no

**Warning!** File rolling by size or time is available only for RealAudio and RealVideo clips encoded using the command-line application. Using DestinationRollSize with any other clip format causes an error.

### File Naming for Rolled Files

File rolling creates multiple files for large outputs. A value of 30, for example, creates a new output clip when the current clip reaches 30 Megabytes in size. Each new file is appended with a number. An output destination named movie.rm is saved as movie.rm, movie1.rm, movie2.rm, and so on.

**Note:** File rolling happens automatically if the clip reaches either the RealMedia file size limit of 4 Gigabytes or the maximum size allowed by the operating system.

### DestinationRollTime

Sets the time into the encoding process at which to write to a new RealMedia file.

value:	[h:][m:]s[.xyz] up to 7:00:00.000
default:	0 (no rolling)
required:	no

**For More Information:** See also “File Naming for Rolled Files” on page 31.

### Timing Values

Time values are accurate to the minute, but should include a seconds designation. If a value is omitted, it is assumed to be zero. Here are some sample values:

60	60 seconds
15:00	15 minutes

### Time-Based Rolling Along with Size-Based Rolling

You can specify both a file rolling time and a file rolling size. In this case, the new file is created when the first specified limit is reached.

### Time-Based Rolling Reflects Encoding Time

Time-based rolling follows encoding time rather than the input media’s timeline. A rolling time of 15 minutes means to write a new file after 15 minutes of encoding time.

Rolled clips may therefore vary in their playback time. With a roll time of 15 minutes, each rolled clip might have a playing time from 10 to 20 minutes depending on the encoding speed.

### Filename

Indicates the path and name of the output file.

value:	file name and path (wildcards not allowed)
default:	none
required:	no, if using the graphical application to encode; yes, if using the command-line application to encode

### When a File Name is Required

An output file name is required when encoding through the command-line application. It is optional with the graphical application, however, because you can add the name using the graphical application before you encode.

### Output File Name Based on Input File Name

You can omit the file name value when you encode from an input file. The output then uses the input’s base file name, adding the appropriate extension.

For example, a CBR RealVideo clip encoded from movie.avi becomes movie.rm. A VBR version becomes movie.rmvb.

#### Archiving for Clips with Existing File Names

If you specify the name of clip that exists in the output directory, the existing clip is archived with the title `_archNNN` appended to the base file name. Hence, movie.rm becomes movie\_arch001.rm before the new movie.rm clip is saved. If you encode the output again, the existing movie.rm becomes movie\_arch002.rm. Higher numbers therefore represent newer archives.

#### Name

---

Assigns a handle to the destination.

value:	string
default:	"Destination <i>n</i> " ( <i>n</i> starts at 1 and increments for each destination)
required:	no (recommended if using the graphical application)

**Note:** This is not related to the file name, and is used only by the graphical application to label the output.

#### PluginName

---

Provides the name of the file system plug-in that writes the file. If omitted, RealProducer determines the plug-in to use based on the `OutputType` value.

value:	plug-in name
default:	rn-file-realmedia for RealAudio and RealVideo
required:	no

**For More Information:** See “`OutputType`” on page 30.

#### xsi:type

---

Indicates that the output is written to a file.

value:	FileDestination
default:	none
required:	yes



## PREFILTERS

Using prefilters, RealProducer can process an input before encoding it. Prefilters allow you to resize a video, increase its contrast, and boost audio gain, for example.

### Pre-filter Syntax

A prefilters section can appear within each `<Input>` list. This section defines the audio and video filters that RealProducer applies to the input before encoding it. The prefilters appear within a list defined by `<Prefilters>` and `</Prefilters>` tags (note the plural).

Each prefilter uses `<Pre-filter.../>` and `</Pre-filter>` tags (note the singular) to enclose the prefilter properties. An `xsi:type` attribute within a `<Pre-filter>` tag determines the type of prefilter used. The following example shows the audio gain prefilter:

```
<Input xsi:type="AVFileInput" ...file input information...>
  <Prefilters>
    <Pre-filter xsi:type="AudioGainPre-filter" Enabled="true" Gain="3.230" />
  </Prefilters>
</Input>
```

**For More Information:** See “Inputs Section” on page 16 for background on the `<Input>` section of the job file.

### Available Prefilters

The following table summarizes the available audio and video prefilters.

**Available Audio and Video Prefilters**

Pre-filter	Function	Reference
Audio Delay Compensation	Corrects audio that is not synchronized with the video.	page 48

(Table Page 1 of 2)

**Available Audio and Video Prefilters (continued)**

Prefilter	Function	Reference
Audio Gain	Amplifies or attenuates the audio signal.	page 47
Black Level Correction	Improves visual contrast and increases the encoding efficiency.	page 42
Cropping	Crops out portions of the input video.	page 37
De-interlace and Inverse-Telecine	Removes unnecessary frames from NTSC videos and corrects line shifting in large videos.	page 39
Resize	Resizes the input video and corrects pixel aspect ratios.	page 43
Video Noise Reduction	Reduces artifacts in the input video.	page 41

(Table Page 2 of 2)

**Tip:** Each prefilter has an Enabled attribute that enables or disables it. This allows you to define a prefilter within a job without activating it. If you think that you may want to use a certain prefilter, add the prefilter syntax to the job file but leave the filter disabled until you need it.

## Prefilter Order for Video Input

RealProducer applies prefilters in the order that they are specified within the job file. Because a prefilter alters video data before the data is passed to the next filter, specifying prefilters in the wrong order can degrade the output. For example, reducing noise can make a video impossible to de-interlace. The following is the recommended order for using prefilters on video input:

1. Cropping
2. Inverse-Telecine
3. De-interlace
4. Video Noise Reduction
5. Black Level Correction
6. Resize

**Tip:** You can apply audio prefilters at any point in the filtering chain.

## Video Input Cropping

You can use the input cropping filter to cut out unnecessary parts of a video, such as the black bars at the top and bottom of a letterboxed, widescreen clip. The following are attributes used with the input cropping filter.

Enabled	Enables the filter.
Height	Gives the total height of the cropped video in pixels.
Left	Sets the number of pixels from the left to start the cropping.
PluginName	Indicates the plug-in to use to perform the cropping.
Top	Sets the number of pixels from the top to start the cropping.
Width	Specifies the total width of the cropped video in pixels.
xsi:type	Indicates the filter type using the value VideoInputCroppingPrefilter.

**Note:** The cropping prefilter does not scale the video larger or smaller. For information about resizing, see “Video Resizing” on page 43.

**Tip:** If you perform a horizontal or vertical resizing on the video first, the cropping offsets are measured from the video’s new size, not its original size.

### Input Cropping Example

The following example creates an output video that is 320 pixels wide by 240 pixels high. The top border begins 12 pixels down from the top border of the input video. The left border begins 24 pixels in from the left border of the input video:

```
<Prefilter xsi:type="VideoInputCroppingPrefilter" Enabled="true"
  Top="12" Left="24" Width="320" Height="240" />
```

### Input Cropping Filter Attributes

The following are the video cropping properties.

#### Enabled

Enables the video cropping filter.

value:	true false
--------	------------

default:	true
required:	no

### Height

Sets the total height of the cropped video in pixels, measured from the point set by the Top attribute.

value:	integer from 32 to 1280
default:	height of input minus the value of Top
required:	no

**Note:** If the value is not a multiple of 4, the next lower multiple is used. For example, a value of 127 results in a video 124 pixels high.

### Left

Defines the number of pixels from the left of the video to start the cropping.

value:	integer from 0 to 2048
default:	0
required:	no

**Tip:** You can specify up to 32 pixels less than the total width. If a video is 240 pixels wide, for example, the Left attribute has a maximum value of 208.

### PluginName

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-inputcropping
required:	no

### Top

Determines the number of pixels from the top of the video to start the cropping.

value:	integer from 0 to 1280
default:	0
required:	no

**Tip:** You can specify up to 32 pixels less than the total height.

### Width

Sets the total width of the cropped video in pixels, measured from the value set by the Left property.

value:	integer from 32 to 2048
default:	width of input minus the value of Left
required:	no

**Note:** If the value is not a multiple of 4, the next lower multiple is used. For example, a value of 162 results in a video 160 pixels wide.

### xsi:type

Indicates the filter to use.

value:	VideoInputCroppingPrefilter
default:	none
required:	yes

## De-Interlace and Inverse-Telecine

De-interlacing is useful for videos larger than 240 pixels high. The inverse-telecine process is for NTSC videos that have been transferred from film. One or both actions can be applied using the same prefilter, which accepts the following attributes:

Deinterlace	Applies the de-interlace filter manually.
Enabled	Enables the filter.
InverseTelecine	Applies the inverse-telecine filter manually.
Manual	Applies the filters only if needed when set to false.
PluginName	Sets the plug-in to use.
xsi:type	Indicates the filter type using the value VideoDeinterlacePrefilter.

### De-Interlace and Inverse-Telecine Examples

The following example sets RealProducer to apply the de-interlace and inverse telecine operations to the input only if it determines they are needed:

```
<Prefilter xsi:type="VideDeinterlacePrefilter" Enabled="true" Manual="false" />
```

The following example applies the de-interlace filter to the input manually:

```
<Prefilter xsi:type="VideDeinterlacePrefilter" Enabled="true" Manual="true"
  Deinterlace="true" />
```

## De-Interlace and Inverse-Telecine Filter Attributes

The following are the valid prefilter attributes.

### Deinterlace

---

Applies the de-interlace filter when the value is set to true.

value:	true false
default:	false
required:	no

**Note:** This is used only if Manual is also set to true.

### Enabled

---

Enables the inverse-telecine filter for automatic or manual operation.

value:	true false
default:	true
required:	no

### InverseTelecine

---

Applies the inverse-telecine filter when the value is set to true.

value:	true false
default:	false
required:	no

**Note:** This is used only if Manual is also set to true.

### Manual

---

When set to false, applies the de-interlace and inverse-telecine filters only when RealProducer determines that they are needed.

value:	true false
--------	------------

default:	false
required:	no

**Tip:** Setting Manual to the default value of false allows you to leave the filter enabled for all content and have RealProducer use it only when necessary.

### PluginName

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-deinterlace
required:	no

### xsi:type

Indicates the filter to use.

value:	VideoDeinterlacePrefilter
default:	none
required:	yes

## Video Noise Reduction

This filter can reduce video noise using a high or low setting. The following are the noise reduction filter attributes:

Enabled	Enables the filter.
Level	Indicates whether to use low or high noise filtering.
PluginName	Sets the plug-in to use.
xsi:type	Indicates the filter using the value VideoNoiseReductionPrefilter.

### Noise Reduction Example

The following is sample syntax for using the low noise reduction prefilter:

```
<Prefilter xsi:type="VideoNoiseReductionPrefilter" Enabled="true" Level="low"/>
```

### Noise Reduction Filter Attributes

The following are the noise reduction filter properties.

### Enabled

---

Enables the noise reduction filter.

value:	true false
default:	true
required:	no

### Level

---

Indicates whether to use low or high noise filtering.

value:	low high
default:	low
required:	no

### PluginName

---

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-videonisereduction
required:	no

### xsi:type

---

Indicates the filter to use.

value:	VideoNoiseReductionPrefilter
default:	none
required:	yes

## Black Level

This filter increases the video contrast and improves the codec efficiency. The following are the available attributes:

Enabled	Enables the filter.
PluginName	Sets the plug-in to use.
xsi:type	Indicates the filter type using the value VideoBlackLevelPrefilter.

## Black-Level Example

The following is sample syntax for the black-level correction prefilter:

```
<Prefilter xsi:type="BlackLevelPrefilter" Enabled="true" />
```

## Black-Level Filter Attributes

The following are the available filter attributes.

### Enabled

Enables the black-level filter.

value:	true false
default:	true
required:	no

### PluginName

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-blacklevel
required:	no

### xsi:type

Indicates the filter to use.

value:	VideoBlackLevelPrefilter
default:	none
required:	yes

## Video Resizing

The video resize prefilter resizes the video input before encoding. The prefilter uses the following attributes:

Enabled	Enables the filter.
PluginName	Sets the plug-in to use.
VideoResizeHeight	Sets the video height in pixels.
VideoResizeQuality	Determines the type of resizing to perform.

VideoResizeWidth	Sets the video width in pixels.
xsi:type	Indicates the filter type using the value VideoResizePrefilter.

**Note:** You can encode each output at a different size. To do this, use the resizing prefilter to set the input to the dimensions of the largest output. Then scale the remaining outputs smaller as needed using the video stream properties. For details, see “Video Stream Properties” on page 59.

**Tip:** If the input video has a non-square pixel aspect ratio (such as DV), you can use the resizing prefilter to shorten the video horizontally. This converts non-square pixels to square pixels.

**Warning!** You should run the resizing filter last, as noted in “Prefilter Order for Video Input” on page 36, because vertical resizing can adversely affect the use of other prefilters.

## Video Resizing Examples

The following example reduces the video size to 176 pixels wide by 132 pixels high:

```
<Prefilter xsi:type="VideoResizePrefilter" Enabled="true"
  VideoResizeWidth="176" VideoResizeHeight="132"
  VideoResizeQuality="high" />
```

### Two-Stage Resizing Example

If you are running the de-interlace filter on a large video, you may be able to decrease the encoding time by resizing the video width, running the other prefilters, then resizing the video height. The initial, horizontal resize decreases the amount of video data that other prefilters must process. Only vertical resizing needs to be performed after other prefilter operations because it can affect de-interlacing. Here is an example:

```
<Prefilters>
  <Prefilter xsi:type="VideoResizePrefilter" Enabled="true"
    VideoResizeWidth="176" VideoResizeHeight="240"
    VideoResizeQuality="high" />
  ...other prefilters, such as de-interlacing...
```

```

    <Prefilter xsi:type="VideoResizePrefilter" Enabled="true"
      VideoResizeWidth="176" VideoResizeHeight="132"
      VideoResizeQuality="high" />
  </Prefilters>

```

The preceding example resizes a video from 320x240 pixels to 176x132 pixels in two steps. In the first resizing operation, the width is reduced. Here, the original height is specified. This keeps the height from being scaled down to maintain the video aspect ratio. In the second resizing, the width is held constant while the height is reduced.

## Video Resizing Filter Attributes

The following are the video resizing attributes.

### Enabled

---

Enables the filter.

value:	true false
default:	true
required:	no

### PluginName

---

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-videoresize
required:	no

### VideoResizeHeight

---

Sets the new video height in pixels.

value:	integer from 32 to 1280
default:	0 (maintains input aspect ratio with the VideoResizeWidth value)
required:	yes, if VideoResizeWidth is not specified

**For More Information:** See “Width and Height Values for Resizing” on page 46.

### VideoResizeQuality

---

Determines the type of resizing to perform.

value:	fast high
default:	high
required:	no

**Tip:** The high value results in better quality, but uses more CPU. It is not recommended for live broadcasts.

### VideoResizeWidth

---

Sets the video width in pixels.

value:	integer from 32 to 2048
default:	0 (maintains input aspect ratio with the VideoResizeHeight value)
required:	yes, if VideoResizeHeight is not specified

**For More Information:** See “Width and Height Values for Resizing” on page 46.

### xsi:type

---

Indicates the filter to use.

value:	VideoResizePrefilter
default:	none
required:	yes

## Width and Height Values for Resizing

For any video resizing attribute, the value must be at least 32. The maximum height is 1280 pixels, and the maximum width is 2048 pixels. Values are automatically rounded down to a multiple of 4, such as 120, 240, 360, and so on. For example, a value of 123 is rounded down to 120. If both height and width are set to 0, no resizing occurs.

To preserve an input video’s aspect ratio, set the height or width to the desired size, then set the other value to 0 (the default). If the input video is 640 pixels wide by 480 pixels high, for example, you can set the width to 320 and the height to 0. This automatically scales the height to 240 pixels.

**Warning!** If you do not preserve the input video's height-to-width ratio when resizing, the output video will appear distorted.

## Audio Gain

The audio gain prefilter amplifies or attenuates the input audio. It uses the following attributes:

Enabled	Enables the filter.
Gain	Sets the audio gain level in dB.
PluginName	Sets the plug-in to use.
xsi:type	Indicates the filter type using the value <code>AudioGainPrefilter</code> .

**Warning!** The audio gain prefilter does **not** support multichannel audio and should not be used when encoding multichannel input.

## Audio Gain Example

The following example demonstrates the audio gain filter syntax:

```
<Prefilter xsi:type="AudioGainPrefilter" Enabled="true" Gain="5" />
```

## Audio Gain Prefilter Attributes

The following are the audio gain attributes.

### Enabled

---

Enables the filter.

value:	true false
default:	true
required:	no

### Gain

---

Sets the audio gain level in decibels (dB). Use a minus sign to indicate negative values.

value:	signed integer from -60 to 60 (-12 to 12 effective range)
--------	---

default:	0
required:	yes

**Tip:** A low value such as -60 effectively mutes the video. High values do not introduce clipping.

### PluginName

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-audiogain
required:	no

### xsi:type

Indicates the filter to use.

value:	AudioGainPrefilter
default:	none
required:	yes

## Audio Delay Compensation

The audio delay compensation prefilter can synchronize audio to the video track. It shifts the audio by the specified number of seconds, eliminating the need to correct the delay through video preprocessing. This filter takes the following attributes:

Advance	Sets the time by which to advance the audio stream.
Delay	Sets the time by which to delay the audio stream.
Enabled	Enables the filter.
PluginName	Sets the plug-in to use.
xsi:type	Indicates the filter type using the value AudioDelayCompPrefilter.

**Tip:** Some video editing tools provide information about how much delay exists between audio and video.

**For More Information:** This filter is useful when you encode audio and video from separate, parallel inputs, as explained in the section “Multiple Inputs” on page 17.

## Audio Delay Compensation Example

The following example demonstrates the audio gain filter syntax:

```
<Prefilter xsi:type="AudioDelayCompPrefilter" Enabled="true" Delay="0.12" />
```

## Audio Delay Compensation Filter Attributes

The following are the filter attributes.

### Advance

---

The time by which to advance the audio stream if it runs behind the video. For example, `Advance="0.3"` advances the audio 3/10ths of one second.

value:	s[.xyz]
default:	0
required:	yes, if Delay is not used.

### Delay

---

The time by which to delay the audio stream if it runs ahead of the video. For example, `Delay="1.1"` delays the audio one-and-1/10th of a second.

value:	s[.xyz]
default:	0
required:	yes, if Advance is not used.

**Tip:** You typically specify either Advance or Delay. If you provide both, the applied value is the difference between the two. For example, if you set Advance to 0.2 and Delay to 0.5, the applied value is a delay of 0.3 seconds.

### Enabled

---

Enables the filter.

value:	true false
default:	true
required:	no

### PluginName

---

Sets the plug-in to use.

value:	filter plug-in name
default:	rn-prefilter-audiodelaycomp
required:	no

### xsi:type

---

Indicates the filter to use.

value:	AudioDelayCompPrefilter
default:	none
required:	yes

## AUDIENCE FILE

An output's audiences determine how the output is encoded. The information in this chapter allows you to edit audience information without using the RealProducer graphical application.

**Note:** You should be familiar with XML syntax conventions such as namespaces, elements, attributes, and values before you edit an audience file manually.

### Audience Section

An audience setting defines audio and video properties for each output. For example, an audience might specify a streaming bit rate of 100 Kbps using the RealVideo10 codec along with a RealAudio soundtrack.

### Audiences in Job Files

In a job file, audiences are defined within an `<Output>` section, as shown in the section “Job File Structure” on page 8. Several elements provide the audience mark-up:

- `<Audiences>...</Audiences>`

The audience section for each output uses a container `<Audiences>` element (note the plural) to encapsulate all audience settings.

- `<Audience>...</Audience>`

Within an `<Audiences>` container, one or more `<Audience>` elements (note the singular) defines each audience.

**Tip:** A single audience creates a single-rate output. Add multiple audiences to an output to create a multi-rate stream.

- `<Streams>...</Streams>`

An `<Audience>` element uses a `<Streams>` element (note the plural) as a container that encapsulates the video and audio stream definitions.

- `<Stream>...</Stream>`

Within the `<Streams>` element, each `<Stream>` element (note the singular) sets the encoding properties for either an audio or a video stream.

## Audience Templates

Audience templates record frequently-used audience settings. They use the file extension `.rpad`, and are stored in the RealProducer audiences subdirectory. RealProducer provides a number of predefined audience templates that you can edit or use as the basis for creating new audience templates.

**Tip:** You can change the location where templates are stored. See “File Path Preferences” on page 155.

### Audience Template Syntax

The syntax for defining an audience within an `.rpad` file is nearly identical to defining an audience within a job file. Unlike a job file, however, an audience file defines only one audience. It does not therefore use the `<Audiences>` container element.

Additionally, the audience file requires an XML declaration and namespaces in the `<Audience>` tag. The following example shows a video and audio stream defined within a stand-alone audience file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Audience
  xmlns="http://ns.real.com/tools/audience.3.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ns.real.com/tools/audience.3.0
http://ns.real.com/tools/audience.3.0.xsd">
  <Streams>
    <Stream xsi:type="AudioStream" ...audio stream information.../>
    <Stream xsi:type="VideoStream" ...video stream information.../>
  </Streams>
</Audience>
```

**Warning!** If you copy the settings from an audience file into a job file manually, be sure to remove the namespaces from the

<Audience> tag. The job file requires only the namespaces defined within the <Job> tag.

**Tip:** The RealProducer graphical application allows you to assign audiences to jobs, modify audience settings, and create new templates. For instructions, refer to *RealProducer User's Guide*.

### Audience File 3.0 Format

RealProducer 13.1 uses the audience file 3.0 format. Older audience files are updated automatically to the 3.0 format when you open them in the RealProducer graphical application.

**Note:** The earlier, 2.0 audience format supported up to four audio profiles for each audience. In the 3.0 format, each audience supports only one audio stream.

## Audience Element

An <Audience> element encapsulates the audio and video stream encoding information for a job or audience template. The <Audience> element can set the following attributes:

LatencyMode	Adjusts system latency in a RealMedia broadcast.
MaxPacketDuration	Sets longest time allowed before writing a packet.
MaxPacketInterleavingDuration	Sets the superblock size for RealAudio streams.
MaxPacketSize	Sets maximum size for an encoded packet.
Name	Provides an audience name.

### Audience Example

The following example illustrates an audio stream and a video stream defined for each of the two audiences used for an output in a job file:

```
<Audiences>
  <Audience Name="Low-Speed Audience">
    <Streams>
      <Stream xsi:type="AudioStream" ...audio stream information.../>
      <Stream xsi:type="VideoStream" ...video stream information.../>
    </Streams>
  </Audience>
```

```

<Audience Name="High-Speed Audience">
  <Streams>
    <Stream xsi:type="AudioStream" ...audio stream information.../>
    <Stream xsi:type="VideoStream" ...video stream information.../>
  </Streams>
</Audience>
</Audiences>

```

## Audience Properties

You can include the following attributes within an <Audience> element.

### LatencyMode

Reduces latency for live broadcasts in the RealMedia format. Ignored for a clip destination.

value:	0 1 2
default:	0 (normal latency, highest error resiliency)
required:	no

**Tip:** You can also set the latency mode at the output level (see “Output Attributes” on page 28). If both values are defined, the audience setting overrides the output setting.

### Available Latency Modes

The LatencyMode property can decrease the end-to-end latency for live broadcasts in the RealMedia format:

- 0 – Normal system latency with high error resiliency.
- 1 – Lower system latency with lower resiliency for dropped packets.
- 2 – Lowest system latency with lowest resiliency for dropped packets.

**Warning!** Use a low latency setting only when delivering a live stream to Helix Server version 11 or later. Earlier versions of Helix Server cannot handle the stream properly.

**For More Information:** See “Latency Mode and Maximum Packet Duration” on page 71.

### MaxPacketDuration

---

Defines the maximum time that RealProducer can encode a RealAudio or RealVideo stream before writing a packet.

value:	s[.xyz] up to 60.000
default:	0 (RealProducer determines when to write packets)
required:	no

**For More Information:** See “Packetizers Section” on page 67.

### MaxPacketInterleavingDuration

---

Sets the superblock size for RealAudio packetization.

value:	s[.xyz] up to 60.000
default:	codec default
required:	no

**For More Information:** See “Maximum Superblock Size” on page 70.

### MaxPacketSize

---

Sets the maximum size for a RealAudio or RealVideo packet, in bytes.

value:	integer up to 15000
default:	600, if bit rate is less than 128 Kbps 1000, if bit rate is 128 Kbps to 256 Kbps 1352, if bit rate is greater than 256 Kbps
required:	no

**For More Information:** See “Packetizers Section” on page 67.

### Name

---

Assigns a unique name to the audience. This is used only by the graphical application to identify the audience.

value:	unique, user-defined string
default:	"Audience n" (n starts at 1 and increments for each audience)
required:	no (recommended if using the graphical application)

**Note:** If multiple audiences use the same name, RealProducer modifies the names to enforce uniqueness.

## Audio Stream Properties

Each audio stream is defined by a <Stream> element. The following are the audio stream properties:

AvgBitrate	Sets the audio bit rate in bits per second (bps).
Channels	Specifies the number of audio channels encoded.
CodecName	Defines the type of codec used to encode the stream.
EncodingComplexity	Sets the audio stream encoding complexity.
PluginName	Specifies the plug-in to use.
xsi:type	Identifies the stream as audio using the value <code>AudioStream</code> .

**Note:** Each <Stream> element falls under a <Streams> container element. See “Audiences in Job Files” on page 51.

## Audio Stream Examples

The following examples illustrate the audience file syntax for audio streams.

### Low-Speed Encoding

The following example uses RealAudio to encode a low-speed, voice-only stream:

```
<Stream xsi:type="AudioStream" AvgBitrate="16000" Channels="1"
  CodecName="ra-voice" />
```

### High-Speed Encoding

The following example shows an RealAudio stream defined for high-bandwidth streaming:

```
<Stream xsi:type="AudioStream" AvgBitrate="64000" Channels="2"
  CodecName="ra" />
```

## Audio Stream Attributes

The following attributes define the audio stream settings for an audience.

### AvgBitrate

Defines the audio bit rate in bits per second (bps).

value:	integer up to 1048576 (1 million bps)
--------	---------------------------------------

default:	none
required:	yes

**Tip:** The average bit rate for the audience is the sum of the audio AvgBitrate value plus the video AvgBitrate value.

**Warning!** You can set audio codecs to specific bit rates only. Setting an unsupported bit rate causes an error. Refer to *RealProducer User's Guide* for information about audio rates.

## Channels

Defines the number of audio channels to encode.

value:	1 2 2s 5.1
default:	none
required:	yes

Possible values mean the following:

1	mono
2	stereo
2s	stereo surround
5.1	multichannel

**Note:** You cannot increase the number of channels provided by the input audio. For example, if the input is standard stereo, encoding as stereo surround or multichannel does **not** add the data necessary to create surround-sound or multichannel effects.

**For More Information:** Refer to *RealProducer User's Guide* for information about channel support.

## CodecName

Defines the type of codec used to encode the stream.

value:	ra ra-voice ra-hr
default:	none
required:	yes

### Available Audio Codec Choices

Codec values are the following.

ra	RealAudio Standard
ra-hr	RealAudio High-Response
ra-voice	RealAudio Voice-Only

**For More Information:** Refer to *RealProducer User's Guide* for information about codec types.

### EncodingComplexity

Sets the encoding complexity for certain audio codecs.

value:	very-low low medium high very-high
default:	high
required:	no

**Tip:** Encoding complexity has little effect on most audio codecs. In general, you can leave this set to the default value.

### PluginName

Provides the name of the plug-in used to encode the audio. If omitted, RealProducer chooses the codec based on the CodecName value.

value:	rn-audiocodec-realaudio
default:	rn-audiocodec-realaudio
required:	no

### xsi:type

Indicates that the stream is audio.

value:	AudioStream
default:	none
required:	yes

## Video Stream Properties

Each video stream is defined by a <Stream> element within the audience. A video stream element can accept the following attributes:

AvgBitrate	Defines the audience average bit rate in bits per second (bps).
CodecName	Indicates the video codec used.
EnableLossProtection	Adds error correction packets.
EncodingComplexity	Sets the encoding complexity.
FramerateMode	Affects RealVideo encoding priority.
MaxBitrate	Sets the maximum bit rate in bits per second (bps).
MaxFramerate	Sets the maximum number of frames per second.
MaxKeyFrameInterval	Sets the maximum time between video key frames.
MaxVideoBufferTime	Sets the maximum clip buffering time in seconds.
OutputHeight	Resizes the output to this pixel height.
OutputWidth	Resizes the output to this pixel width.
PluginName	Defines the plug-in that encodes the stream.
Quality	Sets a quality target for VBR clips.
RateControlMethod	Sets CBR or VBR encoding.
ResizeQuality	Indicates a fast or high-quality resizing.
xsi:type	Identifies the stream as video using the value VideoStream.

**Note:** Each <Stream> element falls under a <Streams> container element. See “Audiences in Job Files” on page 51.

## Video Stream Example

The following examples illustrate the syntax for video streams.

### Low-Speed CBR Encoding

The following example shows a RealVideo 8 stream defined for a low-speed, constant bandwidth. A maximum frame rate of 12 frames per second is set:

```
<Stream xsi:type="VideoStream" AvgBitrate="55000" CodecName="rv8"
  EncodingComplexity="medium" RateControlMethod="CBR"
  MaxFramerate="12.0" />
```

## High-Speed RealVideo VBR Encoding

The following example shows a RealVideo 10 video stream defined for high-bandwidth download or streaming:

```
<Stream xsi:type="VideoStream" AvgBitrate="225000" CodecName="rv10"  
  EncodingComplexity="high" RateControlMethod="VBRUnconstrainedBitrate"  
  MaxFramerate="30.0" FramerateMode="smooth" Quality="90" />
```

## Video Stream Attributes

The following attributes define the video stream settings for an audience.

### AvgBitrate

---

Defines the average bit rate for the video stream in bits per second (bps).

value:	integer to 16777216 (16 Mbps)
default:	none
required:	yes

**Tip:** The average bit rate of the audience is the sum of the audio AvgBitrate value plus the video AvgBitrate value.

**Warning!** If RateControlMethod is set to VBRBitrate, the AvgBitrate setting **must** be less than the value for MaxBitrate.

### CodecName

---

Defines the specific video codec used.

value:	rv8 rv9 rv10
default:	none
required:	yes

**Warning!** In rate-shifting output, all video streams **must** use the same video codec. You cannot encode some streams of a rate-shifting clip as RealVideo 10, for example, and the remaining streams as RealVideo 8.

### EnableLossProtection

---

Adds error correction packets to RealVideo outputs.

value:	true false
default:	false
required:	no

**Tip:** Enabling this attribute can help Helix Server to correct errors during streaming, especially at low bandwidths. This results in a slightly lower quality of video output, however.

**Note:** This error correction capability is different from the forward error correction packets that RealProducer can add to a broadcast stream delivered to Helix Server.

### EncodingComplexity

---

Sets the encoding complexity.

value:	low medium high
default:	high
required:	no

Higher values for EncodingComplexity produce better-quality output, but require more processing. Hence, higher values increase the time (when encoding files) or CPU power (during live broadcasts) that RealProducer requires to encode the output.

### FramerateMode

---

Sets the video mode.

value:	sharp normal smooth slideshow
default:	normal
required:	no

#### Frame Rate Mode Values

This attribute affects how the RealVideo codec attempts to encode scenes when available bandwidth is tight:

- sharp – Focus on image quality, sacrificing smooth motion if necessary.
- normal – Compromise between image quality and smooth motion.

- smooth – Keep motion smooth, sacrificing image clarity as necessary.
- slideshow – Encode as a slideshow of still frames (recommended only for very large videos encoded for very low bandwidths).

**Tip:** This setting is ignored if RateControlMethod is set to VBRQuality or VBRUnconstrainedQuality.

### MaxBitrate

---

Limits the maximum stream speed for certain encoding types. Value is bits per second (bps).

value:	integer to 16777216 (16 Mbps)
default:	none
required:	no, if RateControlMethod is CBR yes, if RateControlMethod is VBRBitrate or VBRQuality

**Warning!** If RateControlMethod is set to VBRBitrate, the MaxBitrate value must be higher than the value for AvgBitrate.

**For More Information:** See also “Rate Control Method” on page 65.

### MaxFramerate

---

Sets the maximum, target frame rate (fps) for the encoded output. The actual frame rate may vary depending on the encoding bandwidth, video frame size, and quality settings.

value:	double-precision number from 0 to 60.0
default:	30.0
required:	no

### MaxKeyFrameInterval

---

Sets the maximum number of seconds between video key frames.

value:	double-precision number from 0 to 60.0
default:	5
required:	no

**Note:** The actual frequency between key frames in encoded output varies depending on the codec, bandwidth, video quality, and so on.

### MaxVideoBufferTime

Sets the maximum time for initial clip buffering, in seconds.

value:	s[.xyz]
default:	4 for RealVideo content
required:	no

### OutputHeight

Sets the video height in pixels.

value:	integer from 32 to 2048
default:	0 (maintains aspect ratio with OutputWidth value)
required:	no

**Warning!** When you encode multiple audiences for a rate-shifting RealMedia output, the video height must be the same for all audiences.

**For More Information:** See “Width and Height Values for Resizing” on page 46.

### OutputWidth

Sets the video width in pixels.

value:	integer from 32 to 2048
default:	0 (maintains aspect ratio with OutputHeight value)
required:	no

**Warning!** When you encode multiple audiences for a rate-shifting RealMedia output, the video width must be the same for all audiences.

**For More Information:** See “Width and Height Values for Resizing” on page 46.

### PluginName

---

Defines the plug-in that encodes the stream. If omitted, RealProducer selects the plug-in based on the CodecName attribute.

value:	rn-videocodec-realvideo
default:	rn-videocodec-realvideo
required:	no

### Quality

---

Sets a quality target for certain types of encoding. A value of 100 represents the highest possible quality.

value:	integer from 1 to 100
default:	100
required:	yes, if RateControlMethod is VBRQuality or VBRUnconstrainedQuality

**For More Information:** See also “Rate Control Method” on page 65.

### RateControlMethod

---

Sets CBR encoding or a type of VBR encoding.

value:	CBR VBRBitrate VBRUnconstrainedBitrate VBRQuality VBRUnconstrainedQuality
default:	CBR
required:	no

**Warning!** If you are creating rate-shifting output, this value **must** be CBR for the video stream of every audience in the output.

**For More Information:** Refer to “Rate Control Method” on page 65.

### ResizeQuality

---

Determines the type of video resizing to perform. The high value results in better quality, but uses more CPU.

value:	fast high
default:	high
required:	no

### xsi:type

---

Indicates that the stream is video.

value:	VideoStream
default:	none
required:	yes

## Rate Control Method

The RateControlMethod attribute for a video stream uses one of five values. The value you select determines whether RealProducer ignores or adheres to the stream's AvgBitrate, MaxBitrate, and Quality settings.

### CBR

---

This value produces a constant-bit-rate stream.

AvgBitrate:	Stream encoded at a consistent bit rate using this value.
MaxBitrate:	Ignored.
Quality:	Ignored.

**Note:** This is the only setting you can use for a video stream when packaging multiple streams into a rate-shifting clip.

### VBRBitrate

---

This is the standard setting for most variable-bit-rate clips.

AvgBitrate:	Stream encoded so that its overall average bandwidth is close to this value. The observed average may be somewhat higher than this value, however.
MaxBitrate:	Bandwidth for certain sections allowed to reach this maximum.
Quality:	Ignored.

### VBRUnconstrainedBitrate

---

This setting creates a variable-bit-rate stream based on the average bit rate:

AvgBitrate:	Stream may have bandwidth spikes, but the average bit rate will be very close to this value.
MaxBitrate:	Ignored. The maximum rate may reach any value.
Quality:	Ignored.

**Tip:** This is a better choice than VBRBitrate for creating a download clip at a specific file size. The final size will be the AvgBitrate setting multiplied by the clip timeline. For example, a 60-second clip with an average bit rate of 450 Kbps is approximately 27,000 Kbits, or 3.3 Megabytes.

### VBRQuality

---

This setting attempts to reach a quality level without exceeding the maximum bandwidth:

AvgBitrate:	Ignored. The average bit rate will vary based on the video contents and quality. The observed average may equal the bandwidth setting. In most cases, though, it will be lower.
MaxBitrate:	Overall bandwidth allowed to reach, but not exceed, this value.
Quality:	Attempt to maintain the specified quality level for the entire stream. Quality may be constrained by the maximum bandwidth, however.

Use VBRQuality under the following conditions:

- You wish to attempt to maintain quality at a certain level, such as 90%.
- It is **not** necessary to maintain a consistent, average bandwidth.
- It is necessary to stay below a maximum bandwidth.

### VBRUnconstrainedQuality

---

This setting attempts to reach a quality level:

AvgBitrate:	Ignored. The average bit rate will vary based on the video contents and the quality level.
-------------	--

MaxBitrate:	Ignored. The maximum bit rate will vary depending on how difficult it is to maintain the quality setting for each section of the video.
Quality:	Attempt to maintain the specified quality level for the entire stream regardless of bandwidth consumption.

Use VBRUnconstrainedQuality under the following conditions:

- You wish to maintain quality at a consistent level, such as 90%.
- It is **not** necessary to maintain a consistent average bandwidth.
- It is **not** necessary to stay below a maximum bandwidth.

## Packetizers Section

Packetization attributes define how RealProducer writes network packets for RealAudio and RealVideo output. You can change the default packetization settings to reduce the latency of live broadcasts and to optimize the packet size for your network. The following elements of an audience or job file, listed here in order of precedence, can define packetization:

- The <Packetizers> element of an audio or video <Stream> element.
- The <Audience> element (see “Audience Properties” on page 54).
- The <Output> element of a job file (see “Output Attributes” on page 28).

### Packetizer Element

To define packetizer settings for an audio or video stream, you add a <Packetizer> element within a <Packetizers> container element of the stream. A <Packetizer> element can use the following attributes:

MaxPacketDuration	Sets longest time allowed before writing a packet.
MaxPacketInterleavingDuration	Sets the superblock size for RealAudio streams.
MaxPacketSize	Sets the maximum size for an encoded packet.
xsi:type	Specifies audio or video packetization.

### Packetizer Example

The following example shows the syntax for defining an audio stream packetizer:

```

<Stream xsi:type="AudioStream" ...audio stream information...>
  <Packetizers>
    <Packetizer xsi:type="AudioPacketizer" ...audio packetizer information... />
  </Packetizers>
</Stream>

```

## Packetizer Attributes

The following are the properties that you can define for an audio or video packetizer.

### MaxPacketDuration

Defines the maximum time that RealProducer can encode a RealAudio or RealVideo stream before writing a packet.

value:	s[.xyz] up to 60.000
default:	0 (RealProducer determines when to write packets)
required:	no

**For More Information:** See “Maximum Packet Size and Duration” on page 69.

### MaxPacketInterleavingDuration

Sets the superblock size for RealAudio packetization.

value:	s[.xyz] up to 60.000
default:	codec default
required:	no

**For More Information:** See “Maximum Superblock Size” on page 70.

### MaxPacketSize

Sets the maximum size for a RealAudio or RealVideo packet, in bytes.

value:	integer up to 15000
default:	600, if target bit rate is less than 128 Kbps 1000, if bit rate is 128 Kbps to 256 Kbps 1352, if target bit rate is greater than 256 Kbps
required:	no

**For More Information:** See “Maximum Packet Size and Duration” on page 69.

**xsi:type**

Specifies an audio or a video packetizer.

value:	AudioPacketizer VideoPacketizer
default:	none
required:	yes

## Maximum Packet Size and Duration

The `MaxPacketDuration` and `MaxPacketSize` properties set the maximum time in milliseconds or the size in bytes, respectively, that `RealProducer` encodes a `RealAudio` or `RealVideo` stream before creating a packet. There are two main uses for this:

- `MaxPacketSize` helps to optimize packets for transmission across a network.
- `MaxPacketDuration` and `MaxPacketInterleavingDuration` (see “Maximum Superblock Size” on page 70) can lower audio broadcast latency.

### Defining Both Packet Duration and Size

When you use these properties together, the first limit reached determines the packet size. For example, you might specify the following:

```
...MaxPacketSize="1450" MaxPacketDuration="0.500"...
```

The preceding properties instruct `RealProducer` to create a packet for the stream when one of the following criteria is met:

- The encoded stream data reaches 1,450 bytes (approximately 1.4 Kilobytes) in size.
- `RealProducer` has encoded the stream for 500 milliseconds since creating the last packet.

### Notes about Setting the Packet Size

Note the following about setting the audio or video packet size:

- You should not set `MaxPacketSize` higher than the maximum transfer unit (MTU) for your network. If you omit `MaxPacketSize` from the packetizers section, a default value applies.

- If `MaxPacketSize` is larger than half of the stream preroll, RealProducer logs a warning and sets the size to half of the preroll.
- Setting the `MaxPacketDuration` value higher than the video preroll results in an increased latency.
- If the value of `MaxPacketDuration` for an audio stream is larger than `MaxPacketInterleavingDuration`, the maximum packet duration is set to the same value as the interleaving duration.
- Setting a value for `MaxPacketDuration` overrides the packet duration specified in the `LatencyMode` attribute of the job file. Therefore, do not specify `MaxPacketDuration` if you want to use `LatencyMode`.

**For More Information:** See “Latency Mode and Maximum Packet Duration” on page 71.

## Maximum Superblock Size

The `MaxPacketInterleavingDuration` property, which functions only for RealAudio streams, sets the maximum superblock size. Although setting a small superblock size lowers the latency for a live audio broadcast, it makes packet loss in the stream more noticeable to the listener.

Omitting `MaxPacketInterleavingDuration` from the packetizer section varies the superblock size with the codec used to encode the stream. If you add `MaxPacketInterleavingDuration` to the packetizer section, you can set any duration value from 0 to 60.000 seconds. Choosing 0 or a value lower than the video frame duration disables audio interleaving, creating no superblocks.

The following example sets a packet duration of one-quarter second and an interleaving (superblock) duration of two seconds. This means that a superblock contains eight packets and causes approximately two seconds of latency before RealProducer transmits the superblock packets to the broadcast server:

```
...MaxPacketDuration="0.250" MaxPacketInterleavingDuration="2.00"...
```

The next example sets a packet size of 250 milliseconds and turns off superblocking. This creates a lower latency than the preceding example, but creates more noticeable audio gaps if packets are lost:

```
...MaxPacketDuration="0.250" MaxPacketInterleavingDuration="0"...
```

## Latency Mode and Maximum Packet Duration

Both a job output (see “Output Attributes” on page 28) and an audience element (see “Audience Properties” on page 54) can define a `LatencyMode` value to decrease the end-to-end latency for live broadcasts in the RealMedia format:

- 0 – Normal system latency with high error resiliency.
- 1 – Lower system latency with lower resiliency for dropped packets.
- 2 – Lowest system latency with lowest resiliency for dropped packets.

Setting a value for `LatencyMode` automatically changes the default values for `MaxPacketDuration` and `MaxPacketInterleavingDuration` used with the stream. The following table explains how the latency mode setting affects the packetization values.

**Effect of Latency Mode on Packets and Superblocks**

Latency Mode	MaxPacketDuration	MaxPacketInterleaving Duration
0	Default packet size of 23 to 480 milliseconds, depending on the codec.	varies with each codec
1	50 milliseconds (8 packets per superblock)	400 milliseconds
2	100 milliseconds	0 (no superblocks)

**Tip:** If you define explicit values for `MaxPacketDuration` and `MaxPacketInterleavingDuration`, their values override the `LatencyMode` setting.

**Note:** The latency mode has no effect on `MaxPacketSize`.

**For More Information:** Contact your RealNetworks representative for more information about reducing latency in a live broadcast.



## SERVER FILE

This chapter explains the syntax for server destination files, which record settings used to transmit a live stream to a broadcast server.

**Note:** You should be familiar with XML syntax conventions such as namespaces, elements, attributes, and values before you edit a server file manually.

**For More Information:** For details about the different types of broadcasts summarized in this chapter, refer to the broadcasting chapter of *RealProducer User's Guide* and *Helix Server Administration Guide*.

## Server Destinations

A server destination specifies the IP address or DNS name of a server used to broadcast a live stream. It also defines properties required by the server, such as the port used to receive the broadcast data. Server destinations are defined within job files for use with the current job. You can also save server destinations as templates.

### Server Destinations in Job Files

A job file for a live broadcast defines at least one server destination within an <Output> element. A server destination uses the same type of <Destination> element used for file output (see “Destinations” on page 26). Its xsi:type attribute defines the destination as a specific type of broadcast method:

```
<ParOutputs>
  <Output...>
    <Destinations>
      <Destination xsi:type="HelixPushServer" .../>
      ...more output destinations...
    </Destinations>
```

```
    ...additional output information...  
</Output>  
    ...additional outputs...  
</ParOutputs>
```

## Server Destination Template Files

RealProducer stores broadcast templates in server destination files, which use the file extension `.rpsd`. The files reside in the RealProducer servers subdirectory.

If you use the graphical application to pick a server destination template, RealProducer copies the destination information from the server file into the job file. You can then change the server settings in the job without affecting the original server destination template.

**Tip:** RealProducer includes sample server destination files for all broadcast types in the `samples/servers` directory under the main RealProducer installation directory.

**Note:** You can change the location where RealProducer stores server destination files. See “File Path Preferences” on page 155.

## Server Element

A stand-alone server destination template (`.rpsd` file), begins with an XML declaration and a `<Server>` container element that defines the appropriate XML namespaces:

```
<?xml version="1.0" encoding="UTF-8"?>  
<Server xmlns="http://ns.real.com/tools/server.3.0"  
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://ns.real.com/tools/server.3.0  
    http://ns.real.com/tools/server.3.0.xsd">  
    <Destination ...all server destination properties.../>  
</Server>
```

**Note:** A job file does not use the `<Server>` container element or the namespace declarations. If you copy the contents of the server template to a job file manually, copy only the `<Destination>` element.

## Push Server Destinations

A server destination for a push broadcast is defined within the `<Destination>` element of the server file or job file. The `xsi:type` attribute defines the broadcast mode:

```
<Destination xsi:type="HelixPushServer" ...broadcast properties.../>
```

### Push Broadcast Types

There are several types of push broadcasts, in which RealProducer starts the broadcast by "pushing" the broadcast data to the server. Each broadcast type uses a different `xsi:type` value.

#### Helix Push

A Helix Push broadcast uses `xsi:type="HelixPushServer"` in the `<Destination>` element. This broadcast method, previously called "account-based push," requires verification against a user name and password stored in the Helix Server authentication database.

#### Helix Advanced Push

This broadcast method uses `xsi:type="HelixAdvancedPushServer"` in the `<Destination>` element. Previously called "password-only push," this broadcast method is verified against a password defined in the Helix Server receiver configuration.

#### Helix Multicast

This broadcast method, which uses `xsi:type="HelixMulticastPushServer"`, is similar to the advanced push broadcast type, except that it multicasts the stream. Use of this broadcast method requires a multicast-enabled network.

## Push Server Attribute Summary

The following table summarizes the attributes used with the various push server modes. It indicates whether an explicit setting for each attribute is required, optional, or not used ("–") for a specific broadcast type.

**Push Server Attributes**

Attribute	Helix Push	Advanced Push	Multicast
AcceptResends	optional	optional	–
Address	<b>required</b>	<b>required</b>	–
FECOffset	optional	optional	optional
FECPercent	optional	optional	optional
HTTPPort	<b>required</b>	–	–
MulticastAddress	–	–	<b>required</b>
MulticastPort	–	–	<b>required</b>
MulticastTTL	–	–	optional
Name	optional	optional	optional
Password	<b>required</b>	<b>required</b>	<b>required</b>
PluginName	optional	optional	optional
Port	–	<b>required</b>	–
ReconnectInterval	optional	optional	optional
ResendListenAddress	optional	optional	–
SavePassword	optional	optional	optional
StatisticsUpdateInterval	optional	–	–
Streamname	<b>required</b>	<b>required</b>	<b>required</b>
Transport	optional	optional	–
Username	<b>required</b>	–	–
xsi:type	<b>required</b>	<b>required</b>	<b>required</b>

**Push Server Properties**

The following are the possible attributes for the <Destination> element.

**AcceptResends**

Accepts the broadcast server's packet resend requests.

value:	true false
default:	true
required:	no

**Tip:** This is ignored for a multicast or if Transport is set to tcp.

### Address

---

Provides the broadcast server's address.

value:	IPv4 address IPv6 address DNS name
default:	none
required:	yes, for Helix Push and Helix Advanced Push

### FECOffset

---

Offsets redundant packets by the specified number of seconds.

value:	1-30
default:	1
required:	no

**Note:** This is used only if FECPercent is 100.

### FECPercent

---

Allocates a percentage of the stream for error correction packets.

value:	0-50 100
default:	0
required:	no

### HTTPPort

---

Used only for a Helix Push broadcast, this indicates the HTTP port on the broadcast server (typically 80).

value:	port number to 65535
default:	none
required:	yes

**Tip:** RealProducer uses the HTTP port to make the initial connection to Helix Server and authenticate the session. Broadcast data is transmitted to Helix Server ports in the range 50001 to 50050.

### MulticastAddress

---

Indicates the multicast address to use.

value:	IPv4 multicast address IPv6 multicast address
default:	none
required:	yes, for multicasts only

### MulticastPort

---

Used for Helix Multicast Push broadcasts, this indicates the port used on the broadcast server to receive stream data.

value:	1-65535
default:	none
required:	yes

**Tip:** By default, Helix Server can receive data on any port in the range 30001-30020. Refer to *Helix Server Administration Guide* for details.

### MulticastTTL

---

Sets the number of router hops allowed for a multicast.

value:	1-255
default:	16
required:	no

### Name

---

Names the server destination in a job file for use by the graphical application.

value:	user-defined name up to 255 bytes
default:	"Server n" (n starts at 1 and increments for each server)
required:	no

**Note:** In a job file, RealProducer adds the default name if the Name value is not defined. If Name is not defined in the server template, the RealProducer graphical interface uses the template file name.

## Password

---

Validates the connection to the broadcast server.

value:	user-defined password up to 255 bytes
default:	none
required:	yes, for most broadcast types

**Note:** When you use the command-line application to run the broadcast, you must enter the password on the command line as well. See “-sd (server definition file)” on page 125.

**For More Information:** See also “SavePassword” on page 80.

### Passwords for Broadcast Modes

Note the following about how passwords are handled for various broadcast modes:

- Helix Push

The password must match a password defined within the Helix Server authentication database.

- Helix Advanced Push and Helix Multicast Push

The password must match the password entered for the broadcast receiver configuration.

**Note:** For both of these modes, Helix Server may define none as the security type. In this case, leave the Password value blank.

## PluginName

---

Identifies the plug-in that handles the stream connection.

value:	plug-in name
default:	rn-server-rbs
required:	no

## Port

---

Used for Helix Advanced Push broadcasts, this indicates the port defined on the broadcast server to receive stream data.

value:	1-65535
--------	---------

default:	none
required:	yes

**Tip:** By default, Helix Server can receive data on any port in the range 30001-30020. Refer to *Helix Server Administration Guide* for details.

### ReconnectInterval

Indicates the number of seconds to wait before attempting to re-establish a broadcast that has been dropped by the destination server.

value:	0-3600
default:	30
required:	no

**Tip:** Set a value of 0 to turn off the reconnection feature.

### ResendListenAddress

Sets the address that RealProducer uses to listen for packet resend requests from the broadcast server.

value:	IPv4 address IPv6 address
default:	0 (use address of default NIC)
required:	no

**Tip:** For the port used to listen for resend requests, the operating system selects an open port above 1024.

**Note:** This property is ignored for multicasts.

### SavePassword

Allows the graphical and command-line applications to save the password as plain text in the server destination file.

value:	true false
default:	true
required:	no

### StatisticsUpdateInterval

---

Used only with Helix Push broadcasts, this attribute sets the interval in seconds that Helix Server sends broadcast statistics back to RealProducer.

value:	0-3600
default:	30
required:	no

**Tip:** Set this to 0 to turn off statistics reporting.

### Streamname

---

Defines the stream name used in the broadcast URL.

value:	stream name with file extension; path optional
default:	none
required:	yes

**Note:** Include the appropriate file extension.

### Optional Path

The stream name can include a virtual path used for Helix Server features such as archiving and splitting. Include a trailing forward slash, as in news/. The value appears in the URL ahead of the stream name. For example:

news/live.rm

**For More Information:** For information about using the path notation to activate Helix Server features, refer to the chapter on transmitters and receivers in *Helix Server Administration Guide*.

### Transport

---

Defines the transport for broadcast packets.

value:	udp tcp
default:	udp
required:	no

Note the following:

- Helix Push

This setting affects broadcast packet delivery only. The initial connection used for authentication always uses HTTP/TCP.

- Helix Advanced Push and Helix Multicast Push

The transport mode must match the mode defined for the broadcast receiver, which is typically udp/unicast.

### Username

Username in the Helix Server authentication database that validates the broadcast.

value:	user-defined name up to 255 bytes
default:	none
required:	yes, for Helix Push broadcasts only

### xsi:type

Indicates the type of broadcast.

value:	HelixPushServer   HelixAdvancedPushServer   HelixMulticastPushServer   RTPPushServer
default:	none
required:	yes

## Push Server Examples

The following sections provide examples of push broadcasting.

### Helix Push Example

The following example defines a Helix Push broadcast. The Username and Password properties supply log-in credentials for the Helix Server authentication database:

```
<Destination xsi:type="HelixPushServer" Address="helixserver.example.com"
  HTTPPort="80" Username="realmedia-encoder" Password="453ERP098zu"
  Streamname="live.rm" Transport="udp" FECPercent="20"
  ReconnectInterval="30" AcceptResends="true"
  ResendListenAddress="172.23.104.188" StatisticsUpdateInterval="20" />
```

### Helix Advanced Push Example

The following example defines a Helix Advanced Push broadcast. There is no user name, and the log-in password is defined in the Helix Server receiver configuration:

```
<Destination xsi:type="HelixAdvancedPushServer"
  Address="helixserver.example.com" Port="30001" Password="546zRGW23"
  Streamname="live.rm" Transport="udp" FECPercent="20"
  ReconnectInterval="30" AcceptResends="true"
  ResendListenAddress="172.23.104.188" />
```

### Helix Multicast Example

The next example illustrates a server destination file for a multicast. The MulticastAddress property defines the multicast address monitored by all Helix Server receivers:

```
<Destination xsi:type="HelixMulticastPushServer" Password="3542wdf22"
  MulticastAddress="227.0.0.17" MulticastPort="30001" MulticastTTL="16"
  Streamname="live.rm" FECPercent="20" ReconnectInterval="30" />
```

## Pull Server Destination

In a pull broadcast, RealProducer does not transmit the broadcast stream until the broadcast server requests it. The server destination for a pull broadcast is defined within the <Destination> element of the server file or job file. The xsi:type attribute defines the broadcast mode:

```
<Destination xsi:type="HelixPullServer" ...all pull server properties.../>
```

### Pull Server Attribute Summary

The following are the attributes used with a pull broadcast:

EncoderAddress	Sets the address used to listen for pull requests.
EncoderListenPort	Sets the port that is monitored for pull requests.
KeepAliveTimeout	Determines how long to wait to stop the broadcast stream.
Name	Names the server destination.
Password	Validates the broadcast.
PluginName	Identifies the plug-in that handles the stream connection.

SavePassword	Saves the password as plain text in the server destination file.
Streamname	Defines the stream name used in the broadcast URL.
xsi:type	Specifies a pull broadcast using the value HelixPullServer.

## Pull Server Properties

The following table lists the possible pull broadcast properties.

### EncoderAddress

Defines the IP address that RealProducer uses to listen for pull requests.

value:	IPv4 address IPv6 address
default:	0 (default address on RealProducer NIC)
required:	no

**Note:** Do **not** use a DNS name.

### EncoderListenPort

Sets the port that RealProducer monitors for pull requests (typically 3031).

value:	1-65535
default:	none
required:	yes

**Note:** You can use the same port for multiple broadcasts running on the same machine.

### KeepAliveTimeout

Sets the number of seconds that RealProducer waits before stopping the broadcast packets after it receives no more “keep alive” messages from the server.

value:	0-86400
default:	30
required:	no

### Name

---

Names the server destination in a job file for use by the graphical application.

value:	user-defined name up to 255 bytes
default:	"Server n" (n starts at 1 and increments for each server)
required:	no

**Note:** In a job file, RealProducer adds the default name if the Name value is not defined. If Name is not defined in the server template, the RealProducer graphical interface uses the template file name.

### Password

---

Validates the connection to the broadcast server. The password must match the password defined in the Helix Server receiver configuration.

value:	user-defined password up to 255 bytes
default:	none
required:	yes

**Tip:** The receiver may also define none as the security type. In this case, leave the Password value in the server destination file blank.

**Note:** When you use the command-line application to run the broadcast, you must enter the password on the command line as well. See “-sd (server definition file)” on page 125.

**For More Information:** See also “SavePassword” on page 86.

### PluginName

---

Identifies the plug-in that handles the stream connection.

value:	plug-in name
default:	rn-server-rbs
required:	no

### SavePassword

---

Allows the graphical and command-line applications to save the password as plain text in the server destination file.

value:	true false
default:	true
required:	no

### Streamname

---

Defines the stream name used in the broadcast URL.

value:	stream name with file extension; path optional
default:	none
required:	yes

**Note:** Include the appropriate file extension.

### Optional Path

The stream name can include an optional, virtual path used for Helix Server features such as archiving or splitting. Include a trailing forward slash, as in news/. The value appears in the URL ahead of the stream name. For example: news/live.rm

**For More Information:** For information about using the path notation to activate Helix Server features, refer to the chapter on transmitters and receivers in *Helix Server Administration Guide*.

### xsi:type

---

Indicates the type of broadcast.

value:	HelixPullServer
default:	none
required:	yes

## Pull Server Example

The following example defines a pull broadcast. RealProducer listens on the IP address and port defined by EncoderAddress and EncoderListenPort, respectively,

for broadcast requests by Helix Server. The Password property verifies access to the stream.

```
<Destination xsi:type="HelixPullServer" EncoderAddress="172.23.104.188"  
  EncoderListenPort="3031" Password="WEPO342zqd" KeepAliveTimeout="25"/>
```

**Tip:** In a pull broadcast, the Helix Server receiver that sends the pull request specifies many of the broadcast parameters to use, such as whether forward error correction is used. For details, refer to the chapter on transmitters and receivers in *Helix Server Administration Guide*.





## COMMAND LINE

The following chapters explain how to use the RealProducer command-line interface.



## COMMAND-LINE BASICS

This chapter explains how to run the RealProducer command-line application, which can encode clips and deliver live broadcast streams. This application provides the same encoding capabilities as the graphical application.

### Command-Line Application

The command-line application is an executable file included in the RealProducer main installation directory:

- producer.exe on Windows

**Note:** On Windows, the RealProducer installer optionally adds the main installation directory to your Path environment variable. If you selected this option, you can run the application from any directory.

### Running the Application

To use the application, open a command-line prompt on your operating system. On Windows, you can click **Start>Run** and enter `cmd` to launch a command-line window. Run the application by entering the name of the executable file followed by options:

```
producer -option1 value1 -option2 value2 ...additional options...
```

The following sections describe the available command options:

“Job Files” on page 99	Use a job file to set all encoding parameters.
“Inputs” on page 103	Specify a file or a capture source as input.
“Metadata” on page 110	Add metadata to the stream.
“Prefilters” on page 113	Use prefilters to enhance the input.
“Outputs” on page 119	Set up single or multiple outputs.

“File Destinations” on page 121	Write outputs to a file.
“Server Destinations” on page 124	Send streams to a broadcast server.
“Audiences” on page 130	Set the stream encoding parameters.
“Logging Options” on page 135	Select information to log.
“Help Options” on page 139	Display help features.

## Stopping the Application

You can stop an encoding session and save the resulting clip by pressing **Ctrl+c**. With multi-audience clips, some merging time is required before the final output is written. To cancel encoding and delete the output and temporary files, press **Ctrl+Break** on Windows.

**For More Information:** See also “Windows Signaling” on page 96 for information about stopping the encoding process from another application.

## Working Directory and Relative Paths

Your input and output files can be located anywhere on your network. Relative paths listed in job files or entered on the command line are assumed to be relative to the working directory, which is the directory from which you run the command-line application.

**Tip:** It is easiest to run the command-line application from the directory that holds your input clips or job file.

## Output and Temporary Directories

To enable the fastest operation, write the encoded output to the same disk that holds the input files. Preferably, this is the local RealProducer disk. Set the temporary directory to a directory on the same disk used for the output. You can define the temporary directory in the preferences file.

**For More Information:** See “File Path Preferences” on page 155.

## Command-Line Encoding

One method for encoding output is to enter all of the encoding parameters on the command line. Most parameters consist of an option flag followed by a value. For example:

```
producer -i movie.avi -o movie_streaming.rm ...additional options...
```

### Basic Command Options

To encode an output using command-line options, you must minimally do the following:

1. Select an input.
  - a. For a file input, use the `-i` option.
  - b. To capture live input, use `-ac` for audio input or `-vc` and `-vp` for video input.

**For More Information:** See “Inputs” on page 103.

2. Set the output type using the `-ot` option.

**For More Information:** See “`-ot` (output type)” on page 121.

3. Define an output destination.
  - a. To encode to a file, use the `-o` option, which the section “`-o` (output file or directory)” on page 123 describes.
  - b. To broadcast live input, use one of the options described in the section “Server Destinations” on page 124.
4. Choose encoding parameters using either the `-ad` or the `-as` option.

**For More Information:** See “Audiences” on page 130.

### Syntax Notes

Note the following about the command-line application syntax:

- Each option is preceded by a hyphen, as in `-i`.
- Most options require a value that immediately follows the option, separated by a space. A few options do not take values, however.
- If an option value includes spaces, enclose the value string (but not the option flag) in double or single quotation marks.

- If you do not include an option on the command line, the option’s default value applies to the encoding job.
- The order of options on the command line does not matter.
- The command-line application supports cross-platform file naming, as described in the section “File Input” on page 17.
- When you encode multiple outputs, you enclose options specific to an output within braces: {...options...}. See “Output Scopes” on page 119.

## Syntax Changes

If you are familiar with RealProducer version 10 or 11, the following sections summarize the differences in the command-line syntax used with RealProducer 13.1.

### Removed Options

The following options have been removed from the RealProducer command-line application.

#### Removed Command-Line Options

Option	Function	Reason for Deletion
-am	Set audio mode.	Audio mode no longer supported.
-arq	Set resampling quality.	High-quality resampling always used.
-duc	Disabled codec update.	No longer needed.
-da	Disabled audio stream.	Stream disabling can be performed using the job file or other encoding options.
-dv	Disabled video stream.	Stream disabling can be performed using the job file or other encoding options.
-vco	Changed RealVideo codec.	Codec selection can be performed using other encoding options.

## Renamed and Modified Options

The options listed in the following table have been renamed in RealProducer 13.1.

**Renamed and Modified Command-Line Options**

Old Name	New Name	Change	Reference
-ag	-f:alg	Renamed option. Value syntax unchanged.	page 114
-bl	-f:vbl	Renamed option. Value syntax unchanged.	page 114
-cr	-f:vcr	Renamed option. Syntax changed from: <i>Left,Top,Width,Height</i> to: <i>WidthxHeight+Left+Top</i>	page 114
-di	-f:vdi	Renamed option. Value syntax unchanged.	page 115
-eco	-as	Video complexity mode now passed as parameter attached to -as value. For example: <i>VideoValue?EncodingComplexity=medium</i>	page 131
-nf	-f:vnf	Renamed option. Value syntax unchanged.	page 115
-rs, -rq	-f:vrs	Combined both option into a renamed option. New value syntax: <i>WidthxHeight@fast high</i>	page 116
-sp	-sp, -sa, -sm	Performs only a Helix Push (“account-based”) broadcast. Use -sa for Helix Advanced Push (“password-only”) or -sm for Helix Multicast. HTTP port value now required.	page 128
-vm	-as	RealVideo mode now passed as a parameter attached to the -as value. For example: <i>VideoValue?FramerateMode=smooth</i>	page 131

## New Options

The following options are new to RealProducer 13.1.

**New Command-Line Options**

Option	Function	Reference
-as	Specifies audio and video encoding settings.	page 131
-dvw	Disables video tests.	page 136
-iep	Sets the timeline position at which encoding stops.	page 106
-isp	Sets the timeline position where encoding starts.	page 106

(Table Page 1 of 2)

## New Command-Line Options (continued)

Option	Function	Reference
-ot	Sets the output type.	page 121
-sa	Starts a Helix Advanced Push (“password-only”) broadcast.	page 124
-sm	Starts a Helix Multicast broadcast.	page 128

(Table Page 2 of 2)

## Job File Processing

A second method of command-line encoding is to define all of the encoding options in a job file. You can then process the job file using the -j option of the command-line application:

```
producer -j my_job.rpjf ...logging options...
```

The job file specifies the inputs, outputs, and all encoding options. However, the command may include a few additional options related to logging.

**For More Information:** For details about using a job file to encode output, refer to “Job Files” on page 99. For more about the job file syntax, refer to “Job File Overview” on page 7.

## Compatibility with Earlier Job Files

RealProducer 13.1 jobs use the job file 3.0 syntax. However, it is backwards compatible with the job file 2.0 syntax used by the following products:

- RealProducer 10
- RealProducer 11

If you have a job file in the 2.0 format, you can process it from the RealProducer 13.1 command-line application. This runs the job and leaves the job file unchanged.

**Tip:** If you have job files that you run frequently, you can update them to the 3.0 syntax. See “Job File Syntax Update” on page 102.

## Windows Signaling

On Windows, the RealProducer command-line application listens for messages sent to its window handle. When the application starts, it registers a

window handle based on the process ID (PID). You can obtain the PID from the Windows Task Manager. Or you can record the PID using the `-pid` option as shown in the following example:

```
producer ...encoding options... -pid "c:\pid.txt"
```

**For More Information:** See “`-pid` (process ID file)” on page 138.

## Signal Producer Utility on Windows

You can send a stop or cancel request to the application by using a utility available in source (C++) and binary formats (`signalproducer.exe`) under the main RealProducer installation directory:

```
samples\utilities\producer_signal_generator
```

### Signal Producer Syntax

The `signalproducer.exe` utility uses the following syntax, in which the `stop` option saves the output and `cancel` option discards it:

```
signalproducer.exe {-p PID|-P PIDFILE} [-a stop|cancel] [-q]
```

### Signal Producer Example

The following example shows how to send a cancel signal to the command-line application using a PID file:

```
signalproducer.exe -P c:\pid.txt -a cancel
```

You can also pass the PID directly using a lowercase `-p` option, as shown in the following example:

```
signalproducer.exe -p 2096 -a cancel
```

## Return Values on Windows

The command-line application returns a value of 0 if no errors occurred during encoding. Otherwise, it returns a value of 1. DOS stores the return value of a command-line program in an environment variable called `ERRORLEVEL`.

This following example checks the contents of `ERRORLEVEL` and prints an error string if `ERRORLEVEL` has a value of 1 or higher:

```
producer -i movie.avi
IF ERRORLEVEL 1 echo Encoding error occurred.
```



## INPUT OPTIONS

Input options of the command-line application allow you to specify a file or capture input, define metadata, and apply prefilters. You can also specify a job file that defines all of the encoding options.

### Job Files

Job files, which use the file extension `.rpjf`, specify the inputs, outputs, and encoding operations. Job file options available for use on the command line include the following:

<code>-cj</code> (create job file)	Creates a new job file.
<code>-ej</code> (evaluate job file)	Evaluates an existing 3.0 job file.
<code>-j</code> (job file name)	Indicates the job file to use when encoding.
<code>-rp</code> (replace parameter)	Replaces a job file variable with an explicit value.

**For More Information:** See “Job File Overview” on page 7 for information about the job file syntax.

### Job File Options

You can include the following job file options on the command line.

#### `-cj` (create job file)

Saves the command-line options into a new job file.

value:	job file name
default:	<code>input_file.rpjf</code>
scope:	all outputs (outside braces)

This option allows you to capture command-line settings into a job file without encoding any content. You can later run the job file by using the `-j` option. For instance, you may want to create a basic job file using the `-cj`

option, then modify the job file manually before running it on the command line.

**Warning!** Job file properties are not validated until the job runs. You can create an invalid job file by specifying a nonexistent audience, for example. In this case, you do not receive an error until you encode a job using the job file.

#### Path Values for Job Files

The value for the `-cj` option is a file name and path, which can be absolute or relative to your current directory. Examples:

```
-cj NewJob.rpjf
-cj c:\Jobs\NewJob.rpjf
-cj /usr/producer/jobs/NewJob.rpjf
```

#### -ej (evaluate job file)

Checks the job file for errors and validates it against the job file 3.0 XSD.

value:	job file name
default:	none
scope:	all outputs (outside braces)

#### Error Reports

Errors in the job file are printed on the command line. Error messages include the settings advisory message IDs. You can find a list of these messages in the `settings_advisor_cli_with_job.xml` file in the resources subdirectory of the main installation directory.

#### -j (job file name)

Specifies the job file to run on the command line.

value:	job file name and path
default:	none
scope:	all outputs (outside braces)

Use either a relative path from the current directory, or an absolute path. For example:

```
-j MyJob.rpjf
```

**Tip:** If the job file specifies inputs that use relative paths, run the command-line application from the job file directory.

### Batch Job File Execution

You can use a wildcard (\*) in the job file name, or specify only a directory to select all job files in that directory. In this case, RealProducer runs all matching job files sequentially.

### Command-Line Options Compatible with the -j Option

Because a job file specifies the encoding options, you cannot use most of the other command-line options. The following options are allowable, however, along with the -j option:

- -cj (create job file) option (see page 99)
- -daw (disable audio watchdogs) option (see page 135)
- -dvw (disable video watchdogs) option (see page 136)
- -lc (logging category) option (see page 137)
- -pid (process ID file) option (see page 138)

### **-rp (replace parameter)**

Supplies a value that replaces a variable in the job file.

value:	<i>variable=value</i>
default:	none
scope:	all outputs (outside braces)

### Job File Variables

Within a job file, you can set a variable for any attribute value by creating a variable name and enclosing the name within percent signs (%). Suppose that you want to encode two outputs from the same job file using different video codecs. In the job file, you specify a single-word variable as the value of the CodecName attribute within the video stream portion of the audience:

```
<Stream xsi:type="VideoStream" CodecName="%VCodec%" .../>
```

When you run the job file, you use the -rp option to specify the variable value:

```
producer -j myjob.rpjf -rp VCodec=rv8
```

```
...
```

```
producer -j myjob.rpjf -rp VCodec=rv10
```

**Tip:** You can create any number of variables as long as each variable within the job file has a unique name. Set a value for each variable using a separate -rp option.

**Warning!** Job files that contain variables cannot be processed by the RealProducer graphical application.

## Job File Examples

The following examples demonstrate how to create job files and encode output using job files.

### Job File Encoding

The following example shows how to run the job file created in the previous example:

```
producer -j GeneralSettings.rpjf
```

### Batch Job File Encoding

The following command runs all job files that begin with the word movies:

```
producer -j movies*.rpjf
```

The next example runs all job files stored in a specific directory:

```
producer -j c:\producer\media\jobs\
```

### Job File Creation

Using the `-cj` option, you can save the command-line settings to a job file. The following example creates the job file `GeneralSettings.rpjf`. This file specifies `movie.avi` as the job input. It automatically sets the output file extension based on the output file type:

```
producer -cj GeneralSettings.rpjf -i movie.avi ...encoding options...
```

To set the output clip to a different name, you include the `-o` option on the command-line:

```
producer -cj GeneralSettings.rpjf -i movie.avi -o output.rmvb ...encoding options...
```

**For More Information:** For more on defining inputs, refer to “-i (input file or directory)” on page 105. The section “-o (output file or directory)” on page 123 describes outputs.

### Job File Syntax Update

If you have job files in the 2.0 syntax, you can update them to the 3.0 syntax. Read the job using the `-j` option and write a new job file using the `-cj` option:

```
producer -j oldjob.rpjf -cj newjob.rpjf
```

**Tip:** The inputs and outputs specified in the job file do not need to be available when you run the update.

**Note:** The command-line application supports the updating of a single job file at a time. Batch mode is not available.

### Batch Job File Creation for Input Files of a Certain Type

If you batch-encode multiple file inputs, you can specify an output directory with `-cj` to create multiple job files. For example, you might use the following command:

```
producer -cj c:\Jobs\ -i c:\media\videos\*.avi ...encoding options...
```

In this case, RealProducer creates a new job file for each AVI file in the directory `c:\media\videos\`. For example, the job file create for an input file named `movie1.avi` is `c:\Jobs\movie1.rpjf`. The job file created for an input file named `movie2.avi` is `c:\Jobs\movie2.rpjf`, and so on.

### Batch Job File Creation for Input Files of All Allowable Types

You can also specify just the input directory, as shown in the following example:

```
producer -cj c:\Jobs\ -i c:\media\videos\ ...encoding options...
```

Here, RealProducer creates a job file for every media file in an acceptable input format, such as AVI or MPEG, that it finds in the input directory. It ignores files in any non-encodable format, such as `.rm` and `.txt` files.

## Inputs

Input options allow you to specify a file or live capture as the input to be encoded.

<code>-ac</code> (audio capture device ID)	Specifies the audio capture input device ID.
<code>-cs</code> (capture frame size)	Defines the input video dimensions.
<code>-d</code> (capture duration)	Sets a duration for audio or video capture.
<code>-i</code> (input file or directory)	Provides the input file name.
<code>-iep</code> (input end position)	Sets the timeline position at which encoding stops.
<code>-isp</code> (input start position)	Sets the timeline position where encoding starts.
<code>-vc</code> (video capture device ID)	Specifies the video capture input device ID.
<code>-vp</code> (video device port)	Indicates the port of the video capture input.

## Input Options

The following input options can be used on the command line.

### **-ac (audio capture device ID)**

Specifies the audio device ID for the input audio device. Required to encode audio from a live input.

value:	integer   string   device name
default:	none
scope:	all outputs (outside braces)

**Tip:** On Windows, set the audio capture port using the Windows recording control panel.

**Note:** Using this option overrides the `-i` option for a file input.

### Audio Capture Values

You can use one of the following values:

- Integer of value 0 or higher identifying a specific device number:  
-ac 0
- String matching an existing device name:  
-ac "Sound Blaster Live!"
- String with a wildcard (\*) matching an existing device name:  
-ac "Sound Blaster \*"
- UNIX device name or symbolic link to a device name:  
-ac /dev/audio

**For More Information:** Use the `-pd` (print device information) option (see page 139) to list device values for your system.

### **-cs (capture frame size)**

Sets the size of the video capture. Must also use `-vc`.

value:	<i>WidthxHeight</i> in pixels
default:	capture card default
scope:	all outputs (outside braces)

Example:

-cs 320x240

**Tip:** The video codec rounds each dimension down to the nearest 4 pixels. If you specify a height of 183 pixels, for example, the output clip has a height of 180 pixels.

**For More Information:** You can also resize a video output. See the section “-f:vrs (video resize)” on page 116.

### **-d (capture duration)**

Sets the duration that the audio or video capture lasts.

value:	[d:][h:][m:]s[.xyz]
default:	encode input until the job manually stopped
scope:	all outputs (outside braces)

#### Duration Time Format

In the duration format, only the number of seconds must be specified.

Examples:

-d 90	90 seconds
-d 45:00	45 minutes
-d 1:00:00	one hour

### **-i (input file or directory)**

Indicates a digitized file on the network or local drive to be encoded. Required to encode from an existing file.

value:	file name and path
default:	none
scope:	all outputs (outside braces)

**Note:** The input file name is ignored if the -ac (audio capture device ID) option (see page 104) or -vc (video capture device ID) option (see page 106) is specified.

**For More Information:** For information about specifying output file locations and names, see “-o (output file or directory)” on page 123.

**Path Values for Input Files**

For the file path, you can specify either an absolute path or a path relative to the working directory.

- Generic example:  
-i movie1.mpg
- Windows examples:  
-i ..\media\movie1.avi  
-i c:\encoder\media\movie1.avi

**-iep (input end position)**

Specifies the point of the input clip timeline where encoding stops.

value:	[d:][h:][m:]s[.xyz]
default:	encode until clip ends or encoding is stopped manually
scope:	all outputs (outside braces)

**For More Information:** For examples of values, refer to the section “Timing Values” on page 22.

**-isp (input start position)**

Sets the point of the input clip timeline where encoding starts.

value:	[d:][h:][m:]s[.xyz]
default:	encode from clip beginning
scope:	all outputs (outside braces)

**For More Information:** For examples of values, refer to the section “Timing Values” on page 22.

**-vc (video capture device ID)**

Sets the device ID for the input video device. Required to encode video from a live input.

value:	integer string device name
default:	none
scope:	all outputs (outside braces)

**Note:** Using this option overrides the -i option for a file input.

### Video Capture Device ID Values

You can use one of the following values:

- Integer of value 0 or higher identifying a specific device number:  
-vc 1
- String matching an existing device name:  
-vc "Osprey Capture Card 1"
- String with a wildcard (\*) matching an existing device name:  
-vc "Osprey\*"
- UNIX device name or symbolic link to a device name:  
-vc /dev/video3

**For More Information:** Use the -pd (print device information) option (see page 139) to list values for your system.

### -vp (video device port)

Sets the port for the video capture device. Must also use -vc.

value:	integer string device name
default:	capture card default
scope:	all outputs (outside braces)

**Note:** This option is not available with Video for Windows (VFW) devices.

**Tip:** Using the -vp option overrides the -i option for a file input.

### Video Device Port Values

You can use one of the following values:

- Integer of value 0 or higher identifying a port number:  
-vp 1
- String matching an existing port name:  
-vp "S-Video"
- String with a wildcard (\*) matching an existing port name:  
-vp "Composite\*"

**For More Information:** Use the `-pd` (print device information) option (see page 139) to list values for your system.

## File Input and Output Examples

The following examples illustrate the most basic uses of the command-line application.

### Constant Bit Rate Encoding

The following example provides a simple example of encoding of a media clip from an input file:

```
producer -i movie.avi -ad "65k RV" -ot rm
```

Because no output is specified, the encoded clip is named `movie.rm`. The `.rm` extension is used automatically because the `-ot` option sets the output type. A single audience is specified using the `-ad` option.

Optionally, you can save the clip to a different directory or file name by including the `-o` option:

```
producer -i movie.avi -ad "65k RV" -ot rm -o c:\videos\movie_streaming.rm
```

For usage details, refer to the following:

- “`-i` (input file or directory)” on page 105
- “`-ot` (output type)” on page 121
- “`-o` (output file or directory)” on page 123
- “`-ad` (audience file)” on page 130

### Variable Bit Rate Encoding

When using a variable bit rate audience, the output uses the `.rmvb` file extension. You can select only one VBR audience per encoding:

```
producer -i movie.avi -o movie.rmvb -ad "750k RV" -ot rm
```

### Batch Encoding Using Wildcards

For batch encoding, you can specify the directory that contains the source files, or use one or more asterisks (\*) as wildcards in file names. The following example encodes a batch of files using a wildcard to designate all MPEG clips in the specified input directory. The output clips are placed in the `movies` directory:

```
producer -i C:\source\*.mpeg -o C:\movies\ -ad "65k RV" -ot rm
```

**Note:** You cannot use a wildcard in a directory path.

### Writing Outputs to the Input Directory

If you do not specify output paths and names, the encoded clips are written to the input directory, using each input's base file name and the file extension appropriate for the output file type. Examples:

```
-i *.mpg
-i c:\files\media\trailer*.avi
-i /home/encoder/media/*.mpg
```

### Omitting Input and Output File Names

If you specify just the input directory, RealProducer creates an output for every media file in an acceptable input format, such as AVI or MPEG, that it finds in the input directory. It ignores files in any non-encodable format, such as .rm and .txt files. Examples:

```
-i c:\sources\video\
-i ../media/inputs
-i /home/encoder/media/
```

Optionally, you can add the `-o` option to specify a different output directory:

```
-i c:\sources\video\ -o c:\outputs\clips\
-i ../media/video -o ../clips/streaming
-i /home/encoder/media/ -o /home/encoder/output/
```

## Live Capture Examples

The following examples demonstrate how to capture live input and encode it to a file or send it to a server for broadcast.

### Live Input Capture to File

The following example captures input from the video card and the sound card, writing the output to a clip named `capturefile.rm`:

```
producer -vc 0 -ac "Sound Blaster *" -o c:\capturefile.rm -ad "65k RV" -ot rm
```

For details about capture options, refer to:

- “`-ac` (audio capture device ID)” on page 104
- “`-vc` (video capture device ID)” on page 106

## Live Input Capture for Broadcast

The following example captures audio and video, delivering the encoded stream to Helix Server using the Helix Push method of broadcasting. This broadcast method requires a user name (encoder in this example) and password:

```
producer -vc 0 -ac 0 -sp encoder:as45er897@helixserver.example.com:80/news.rm  
-ad "80k RV" -ot rm
```

**For More Information:** See “Server Destinations” on page 124.

## Capture for Multi-Rate Audiences

The following example captures audio and video streams, sending the output to both a server and a file destination. The server broadcast uses the Helix Advanced Push method. The encoding targets the 65 and 80 Kbps audiences:

```
producer -vc 0 -ac 0 -sa TY45poi@helixserver.example.com:30001/sports/live.rm  
-o archive.rm -ad "65k RV,80k RV"
```

## Metadata

Metadata informs the user about the clip or live stream. Adding metadata to the encoded output is highly recommended.

-a (author)	Encodes an author name.
-c (copyright)	Sets the encoded copyright.
-de (description)	Adds a clip description.
-k (keywords)	Adds keywords.
-r (content rating)	Rates the content of the output.
-t (title)	Adds a title.

## Metadata Options

The following command-line options encode metadata into a clip or live stream.

### -a (author)

Indicates the author or owner of the clip or broadcast.

value:	string up to 255 characters
--------	-----------------------------

default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

Example:

-a "Brilliant Media Limited"

### **-c (copyright)**

Defines the copyright owner and date.

value:	string up to 255 characters
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

Example:

-c "2009 Brilliant Media Limited"

### **-de (description)**

Sets a description of the clip that is longer and more informative than the title.

value:	string up to 64 KB
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

Example:

-de "An encore presentation of the 2009 music awards."

### **-k (keywords)**

Encodes keywords that help search engines index the clip.

value:	string up to 255 characters
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

Example:

-k "awards 2009 music"

### **-r (content rating)**

---

Defines the age range for which the content is applicable.

value:	0-6
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

### **RealMedia Ratings**

For RealMedia clips, you can use a numeric rating from 0 to 6.

#### **RealMedia Ratings**

Numeric Rating	Meaning
0	No Rating (this is the default)
1	All Ages
2	Older Children
3	Younger Teens
4	Older Teens (15 and up)
5	Adult Supervision Recommended
6	Adults Only

For example:

```
-r 1
```

### **-t (title)**

---

Defines a title for the clip or broadcast.

value:	string up to 255 characters
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

Example:

```
-t "2009 Music Awards"
```

## **Metadata Examples**

The following examples demonstrate how to encode metadata into streams.

## Basic Clip Information

The following example encodes basic metadata into the clip:

```
producer -i myfile.mov -o output.rm -ad "65k RV" -ot rm
-t "Summer Vacation"
-a "John Doe"
-k "vacation cape cod 2004"
-de "Video from summer vacation in Cape Cod."
```

## Global and Output-Specific Values

This example sets the same title and author for each output clip. Each clip uses a different description, however.

```
producer -i myfile.mov -t "Summer Vacation" -a "John Doe"
{-o output1.rm -de "low-bandwidth version" -ad "65k RV" -ot rm}
{-o output2.rm -de "high-bandwidth version" -ad "300k RV" -ot rm}
```

**For More Information:** Refer to “Outputs” on page 119 for details about multiple outputs.

## Prefilters

Prefilters modify the audio or video input before it is encoded. Because all prefiltering is optional, no prefilters are applied by default.

-f:alg (audio level gain filter)	Adjusts the audio gain.
-f:vbl (black-level filter)	Increases video contrast.
-f:vcr (video cropping)	Sets cropping parameters in pixels.
-f:vdi (inverse-telecine and de-interlace filters)	Applies de-interlace or inverse-telecine filters.
-f:vnf (video noise filter)	Removes video noise.
-f:vrs (video resize)	Resizes the video.

**Note:** You cannot apply the audio compensation prefilter, which adjusts audio that is out of synchronization with video, through the command line. You can specify this prefilter in a job file, however. See “Audio Delay Compensation” on page 48.

## Prefilter Options

The following prefilter options are available for use on the command line.

**-f:alg (audio level gain filter)**

---

Applies the audio gain filter to amplify or attenuate the input audio. This can boost the audio signal or decrease it to prevent it from clipping.

value:	-48.0 (attenuation) to 48.0 (amplification)
default:	no audio change
scope:	all outputs (outside braces)

**Tip:** The effective range for this option is -12 to 12. A large attenuation value, such as -48, effectively mutes the audio.

**-f:vbl (black-level filter)**

---

Increases contrast in videos by making dark colors pure black and off-white colors pure white.

value:	none
default:	do not adjust black level
scope:	all outputs (outside braces) or individual outputs (inside braces)

**-f:vcr (video cropping)**

---

Crops out areas from the sides of the input video.

value:	<i>WidthxHeight+Left+Top</i>
default:	no cropping
scope:	all outputs (outside braces) or individual outputs (inside braces)

**Tip:** As described in “-f:vrs (video resize)” on page 116, you can also resize a video. Because cropping occurs first, the *Width* and *Height* values of the -f:vcr option indicate the input size for the resizing operation.

**Note:** When capturing live video, you can use the -cs (capture frame size) option (see page 104) to set the input frame size. This can eliminate the need to crop or resize the input.

**Input Cropping Values**

The option uses four pixel values:

<i>Width</i>	Width of the cropped video, measured from the point set by the <i>Left</i> property. If the width value is not a multiple of 4, the next lower multiple is used. For example, a value of 162 results in a video 160 pixels wide.
<i>Height</i>	Height of the video, measured from the point set by the <i>Top</i> property. If the value is not a multiple of 4, the next lower multiple is used. For example, a value of 127 results in a video 124 pixels high.
<i>Left</i>	Offset from the left edge where video cropping begins. You can specify up to 32 pixels less than the total width.
<i>Top</i>	Offset from the top edge where video cropping begins. You can specify up to 32 pixels less than the total height.

**Input Cropping Example**

In the following example, the input video is cropped to 320x240 pixels. The cropping starts 10 pixels to the right of the video's left edge and 20 pixels down from the top edge:

```
producer -i movie.avi -f:vcr 320x240+10+20 ...encoding options...
```

**-f:vdi (inverse-telecine and de-interlace filters)**

Specifies the use of the inverse-telecine or de-interlace filter on the video input. The de-interlace filter is generally needed only on videos larger than 320 by 240. The inverse-telecine filter is for videos that were transferred from film.

value:	auto d i both
default:	filter not applied
scope:	all outputs (outside braces) or individual outputs (inside braces)

The following are the values that you can specify:

auto	Autodetect and apply both filters only if needed. This is recommended.
d	Apply the de-interlace filter only.
i	Apply the inverse-telecine filter only.
both	Apply both the de-interlace and inverse-telecine filters.

**-f:vnf (video noise filter)**

Removes small distortions in the video image.

value:	low high
--------	----------

default:	no noise filtering performed
scope:	all outputs (outside braces) or individual outputs (inside braces)

**Tip:** Use a value of low for small distortions, a value of high if the noise is more pronounced. Do not apply these filters to video input that is free of distortion.

### **-f:vrs (video resize)**

Sets the output video's width and height in pixels. Also specifies fast resizing or high-quality resizing

value:	<i>WidthxHeight@fast high</i> ( <i>Width</i> and <i>Height</i> in pixels, 0 to 2048)
default:	no resizing
scope:	all outputs (outside braces) or individual outputs (inside braces)

**Tip:** If you set one dimension to 0, that dimension scales in proportion to the other dimension, maintaining the input clip's aspect ratio.

**Note:** The high value results in a higher-quality output, but requires considerably more processing power during encoding. The fast value uses less processor power, but results in lower quality.

#### **Resizing Examples**

The following are resizing examples:

-f:vrs 320x240@fast	Fast resize to 320 pixels wide by 240 pixels high.
-f:vrs 180x0@fast	Fast resize to a width of 180 pixels. Scale the height proportionally, maintaining the input video's aspect ratio.
-f:vrs 0x160@high	High-quality rresize to a height of 160 pixels. Scale the width proportionally, maintaining the input video's aspect ratio.

**Note:** Video codecs round each dimension down to the nearest 4 pixels. If you specify a height of 183 pixels, for example, the output clip has a height of 180 pixels.

#### **Prefilter Examples**

The following examples show how to modify the input by applying prefilters.

## Multiple Prefilters

This example applies the audio gain prefilter with a value of 4, uses black-level correction, and automatically detects if the de-interlace or inverse-telecine filter needs to be applied:

```
producer -i movie.mov -o movie.rm -ad "65k RV" -f:alg 4 -f:vbl -f:vdi auto
```

## Cropping and Resizing a Single Output

The next example assumes an input in wide-screen format (2.35:1 ratio), with a width of 564 pixels and a height of 240 pixels. First, 102 pixels are cropped from both the left and right sides to create an input that is 360 by 240 (4:3 ratio). Then the video is reduced to approximately 240x180 pixels using a high-quality resize.

```
producer -i widescreen.avi -o normal.rm -ad "65k RV" -ot rm
-f:vcr 320x240+102+0 -f:vrs 240x0@high
```

**Tip:** RealProducer always applies prefilters in the necessary order regardless of the order in which they are entered on the command line. If the resizing command in this example preceded the cropping command, the cropping operation would still occur first.

## Cropping the Input and Resizing Outputs Separately

The following example uses the same cropping parameter as the previous example to fit a widescreen clip into a 4:3 aspect ratio. It produces three separate outputs, however, resized for different bandwidths:

```
producer -i widescreen.avi -f:vcr 320x240+102+0
-o output_high.rm -ad "300k RV" -ot rm}
-o output_medium.rm -f:vrs 240x0@fast -ad "150k RV" -ot rm}
-o output_low.rm -f:vrs 176x0@fast -ad "65k RV" -ot rm}
```

The first output is encoded at 300 Kbps. Because no resizing is specified, it uses the cropped video size of 320x240 pixels. The second output streams at 150 Kbps and is resized to 240x180 pixels. The last output, which streams at 65 Kbps, is resized to 176x132 pixels.

**For More Information:** Refer to “Outputs” on page 119 for details about multiple outputs.



## OUTPUT OPTIONS

This chapter covers the options for defining destinations and audiences when you encode outputs using the command-line application.

### Outputs

An encoding job can have one or more outputs. For example, the same media input might be encoded as two separate outputs:

- A low-bandwidth clip encoded at a single bit rate using RealVideo 8.
- A high-bandwidth clip encoded at a three bit rates using RealVideo 10.

To encode multiple outputs, you enclose the options for each output in braces:

```
producer ...input options... {output 1 options} {output 2 options} ...
```

### Output Scopes

Each command-line option in this document includes a *scope* designation that indicates where the option falls (outside braces, inside braces, or either).

**Tip:** If you are encoding only one output, you do not need to use braces at all, and you can ignore the scope designation.

**Note:** Braces cannot be nested.

### All Outputs

If the scope for a command-line option is listed as *all outputs*, the option must appear *outside* of any sets of braces. For example, the `-i` option to select an input file applies to all encoded outputs:

```
producer -i movie1.avi {output 1 options} {output 2 options} ...
```

## Individual Outputs

If the scope for a command-line option is listed as *individual outputs*, the option must appear *inside* a set of braces when encoding multiple outputs. For example, the `-o` option to specify an output file name applies to each output individually:

```
producer -i movie1.avi {-o file1.rm...} {-o file2.rm...} ...
```

## All Outputs or Individual Outputs

If the scope for a command-line option is listed as *all outputs or individual outputs*, the option can appear either outside or inside of each set of braces. List the option outside of all braces to apply the option to every output.

### Global Value with Individual Overrides

You can also place the option in both locations to create a global value and individual overrides. For example, you can use the `-t` option to define a clip title:

```
producer -i movie1.avi -t "first title" {output 1 options} {-t "second title"...} ...
```

In the preceding example, the global value *"first title"* is encoded into all outputs that do not specifically override the title value, as shown with the second output.

## Output Options

The following options set overall output features.

<code>-lm</code> (latency mode)	Sets the broadcast latency mode.
<code>-mtu</code> (maximum packet size)	Specifies a broadcast's maximum packet size.
<code>-ot</code> (output type)	Sets the output file type.

### `-lm` (latency mode)

Affects the latency mode of a live broadcast in the RealMedia format.

value:	0 1 2
default:	0 (no latency reduction)
scope:	all outputs (outside braces)

**For More Information:** See "Available Latency Modes" on page 28.

**-mtu (maximum packet size)**

Sets the maximum packet size in bytes for live broadcast streams.

value:	bytes
default:	600, if target bit rate is less than 128 Kbps 1000, if target bit rate is between 128 Kbps and 256 Kbps 1352, if target bit rate is greater than 256 Kbps
scope:	all outputs (outside braces)

**For More Information:** See “Latency Mode and Maximum Packet Duration” on page 71.

**-ot (output type)**

Required option that sets the output file type.

value:	ra – RealAudio output with a .ra file extension rm – RealAudio or RealVideo output with a .rm file extension rv – RealVideo output with a .rv file extension
default:	none
scope:	individual outputs (inside braces)

## File Destinations

A file destination encodes the output to a file. The following options are available.

-drs (destination roll size)	Starts a new output file by size.
-drt (destination roll time)	Rolls output files by encoding time.
-dt (disable two-pass)	Disables two-pass encoding.
-o (output file or directory)	Specifies the output file name and path.

**-drs (destination roll size)**

Sets a file size limit. RealMedia clips only.

value:	Megabytes
default:	file format or operating system maximum size
scope:	all outputs (outside braces) or individual outputs (inside braces)

**For More Information:** See “File Destination Attributes” on page 31.

**-drt (destination roll time)**

---

Sets an encoding duration after which RealProducer creates a new output file. RealMedia clips only.

value:	[h:][m:]s[.xyz] up to 7:00:00.000
default:	file format or operating system maximum size
scope:	all outputs (outside braces) or individual outputs (inside braces)

**For More Information:** See “File Destination Attributes” on page 31.

**Example**

The following example creates a new file archive after RealProducer has encoded the output for 15 minutes:

```
-drt 15:00
```

**-dt (disable two-pass)**

---

Makes clip-to-clip encoding occur in a single pass.

value:	no value required
default:	encode clip-to-clip using two-pass
scope:	all outputs (outside braces)

**Tip:** Disabling two-pass encoding lowers the output quality. Do so only if you need to speed clip-to-clip encoding time.

**Automatic Disabling of Two-Pass Encoding**

Two-pass encoding is automatically disabled if you do any of the following:

- Capture live audio using the `-ac` (audio capture device ID) option (see page 104).
- Capture live video using the `-vc` (video capture device ID) option (see page 106).
- Send output to a server destination as described in the section “Server Destinations” on page 124.

**-o (output file or directory)**

Sets the output clip's file name and path to an existing directory.

value:	file name and optional path
default:	input file name with output format file extension
scope:	individual outputs (inside braces)

**Tip:** This option is required when capturing live input. With file-to-file encoding, omitting the `-o` option encodes the clip using the input file's base file name and the file extension appropriate for the output type. The output clip is saved to the directory that holds the input file.

**For More Information:** For examples of setting output file names and directories, see “File Input and Output Examples” on page 108.

**File Name Examples**

The following examples specify output paths and file names:

```
-o C:\Movies\preview1.rm
-o /usr/producer/output/movie2.rmvb
```

**Note:** The path can be absolute or relative to the working directory.

**Multiple Outputs**

When you encode multiple outputs, save each output to a different file name or directory:

```
{-o preview1.rm ...}{-o preview2.rm ...}
```

**Archive Clips**

If you specify the name of clip that exists in the output directory, RealProducer does not overwrite the existing clip. Instead, it archives the existing clip by appending `_archMMM` to the base file name.

For example, `movie.rm` becomes `movie_arch001.rm` before the new `movie.rm` clip is saved. If you encode the output again, the existing `movie.rm` becomes `movie_arch002.rm`. Higher numbers therefore represent newer archives.

## Server Destinations

Server destinations specify a broadcast server that receives a live stream. You can use any of the following options to deliver a stream to a broadcast server. To broadcast to multiple servers, add multiple server destinations to the output stream.

-sa (Helix Advanced Push destination)	Starts a Helix Advanced Push broadcast.
-sd (server definition file)	Runs any type of broadcast using a server destination template.
-si (Helix Pull destination)	Starts encoding in pull mode.
-sm (Helix Multicast Push destination)	Runs a Helix Multicast Push broadcast.
-sp (Helix Push destination)	Creates a Helix Push broadcast.

**Note:** You can define only one push destination for an output. However, if you set up a push destination, you can also define a pull destination for the same output.

**For More Information:** For background about broadcasting destinations, refer to the broadcasting section of *RealProducer User's Guide*.

## Server Destination Options

The following options define broadcast servers as encoding destinations.

### **-sa (Helix Advanced Push destination)**

The -sa option defines a Helix Advanced Push broadcast with Helix Server as the destination.

value:	<i>password@address:port/path/stream</i>
default:	none
scope:	individual outputs (inside braces)

### Helix Advanced Push Destination Components

The following table explains the Helix Advanced Push connection string settings.

<i>password</i>	Password defined on the Helix Server receiver definition. The receiver must be set to use the Basic authentication method.
<i>address</i>	Valid IPv4 address, IPv6 address, or domain name for the Helix Server destination.
<i>port</i>	Server port that receives the stream. The receiver typically accepts connections on ports 30001 through 30020.
<i>path</i>	Optional, virtual path used to define server-side features.
<i>stream</i>	The name of the broadcast stream. Include the appropriate extension, as in live.rm.

**Warning!** The stream name cannot contain the following characters: @ : /

### Helix Advanced Push Broadcast Example

In the following example, RealProducer delivers the stream to Helix Server port 30001. A virtual path used by the server is provided, along with the stream name live.rm:

```
-sa TY45poi@helixserver.example.com:30001/sports/live.rm
```

### **-sd (server definition file)**

The `-sd` option allows you to broadcast in any mode using a predefined server destination.

value:	<i>username:password@definition,path/stream</i>
default:	none
scope:	individual outputs (inside braces)

When you use this option, a server destination file sets the necessary broadcast parameters, such as the server's IP address, transport method, and advanced parameters like error correction.

**For More Information:** See “Server Destinations” on page 73 for an explanation of server files and server destination syntax.

### Server Definition Components

The following are the components for the `-sd` command. The actual values required to run the broadcast vary with the type of broadcast performed.

<i>username</i>	User name required by the broadcasting method. If no user name is required, as with Helix Advanced Push broadcasting, omit this value.
<i>password</i>	Password required by the broadcasting method. If no password is required, omit this value.
<i>definition</i>	Name of a server template stored in the default directory. Or, the path and file name of a server definition file stored anywhere on your network.
<i>path</i>	Optional, virtual path used to define server-side features.
<i>stream</i>	The name of the broadcast stream. Include the appropriate extension, as in <code>live.rm</code> .

**Warning!** The stream name cannot contain the following characters: `@ : /`

### Server Template Example

Server templates are server destination files stored in the `servers` directory under the main RealProducer directory. To use a template, you specify the template name, which is the value of a `Name` property defined within the file.

The following example specifies a template named `Rock Radio Pull Broadcast`. When you select a template, you can abbreviate the name down to a unique set of characters, such as `Rock`:

```
-sd 567dlopf212@Rock,livefeed.rm
```

**For More Information:** Use the `-ps` (print servers) option (see page 139) to display a list of available server templates.

### Server File Example

You can also specify any server destination file on your network by providing the file's path (absolute or relative to the working directory), file name, and extension (`.rpsd`).

The following example specifies the absolute path to a server destination file that sets up a Helix Push broadcast. Both a user name and password are supplied, as required by the Helix Push broadcast method:

```
-sd remote_encoder:as45er897@/usr/encoder/pushbroadcast.rpsd,football.rm
```

The next example provides the relative path to a server destination file that defines a Helix Advanced Push broadcast. Accordingly, the `-sd` option value

supplies just a password. A virtual path of `hourly/` is included with the stream name:

```
-sd yop563sdf@../definitions/SydneyReceiver.rpsd,hourly/news.rm
```

### **-si (Helix Pull destination)**

The `-si` option defines a pull broadcast accessible by Helix Server.

value:	<code>password@listenAddress:listenPort/path/stream</code>
default:	none
scope:	individual outputs (inside braces)

### Helix Pull Destination Components

The following are the components of a Helix Pull broadcast.

<i>password</i>	Password used by RealProducer to authenticate server pull requests. The Helix Server receiver must supply this password. If you omit this, any pull-enabled server can connect to the stream.
<i>listenAddress</i>	Valid version 4 or version 6 IP address to which RealProducer binds. The default value is the default IP address on the default network interface card.
<i>listenPort</i>	Port that RealProducer uses to listen for pull requests.
<i>path</i>	Optional, virtual path used to define server-side features
<i>stream</i>	The name of the broadcast stream. Include the appropriate extension, as in <code>live.rm</code> .

**Warning!** The stream name cannot contain the following characters: `@ : /`

### IP Address Values

To find the listen address of the RealProducer machine on which the command-line application is running, do the following:

- On Windows, open a command prompt and type the following:

```
ipconfig
```

### Pull Broadcast Examples

The following example shows how to start a pull broadcast using the `-si` option and specifying all values, including a virtual path name:

```
-si dkgy435ty@192.168.224.221:3031/rock/radio.rm
```

To use the RealProducer default IP address, you specify a 0 value for the address. For example:

```
-si dkg435ty@0:3031/music.rm
```

### **-sm (Helix Multicast Push destination)**

The `-sm` option defines a Helix Multicast Push broadcast with Helix Server as the destination.

value:	<i>password@address:port/path/stream</i>
default:	none
scope:	individual outputs (inside braces)

### Helix Multicast Push Destination Components

The following table explains the Helix Multicast Push connection string settings.

<i>password</i>	Password defined on the receiver definition. The receiver must be set to use the Basic authentication method.
<i>address</i>	A class-D multicast address.
<i>port</i>	Server port that receives the stream. The Helix Server receiver typically accepts connections on ports 30001 through 30020.
<i>path</i>	Optional, virtual path used to define server-side features.
<i>stream</i>	The name of the broadcast stream. Include the appropriate extension, as in <code>live.rm</code> .

**Warning!** The stream name cannot contain the following characters: @ : /

### Helix Multicast Push Example

In the following example, the stream is multicasted to port 30011. No virtual path is used with the stream name:

```
-sm 569ad42k@224.0.0.1:30011/playoffs.rm
```

### **-sp (Helix Push destination)**

The `-sp` option defines a Helix Push broadcast with Helix Server as the destination.

value:	<i>username:password@address:port/path/stream</i>
default:	none
scope:	individual outputs (inside braces)

### Helix Push Destination Components

The following are the Helix Push connection string settings.

<i>username</i>	User name defined in the Helix Server authentication database.
<i>password</i>	Password defined in the Helix Server authentication database.
<i>address</i>	Valid IPv4 address, IPv6 address, or domain name for the Helix Server destination.
<i>port</i>	Server's HTTP port. Defaults to 80.
<i>path</i>	Optional, virtual path used to define server-side features.
<i>stream</i>	The name of the broadcast stream. Include the appropriate extension, as in live.rm.

**Warning!** The stream name cannot contain the following characters: @ : /

### Helix Push Broadcast Example

In the following example, no virtual path is given, and the stream name is news.rm:

```
-sp remote_encoder:as45er897@helixserver.example.com:8080/news.rm
```

If authentication is not required on the server, you can omit the user name, password, and @ sign:

```
-sp helixserver.example.com:8080/news.rm
```

## Additional Server Parameters

You can append additional parameters to any server destination string by adding query string parameters separated by the “?” and “&” operators. For example, the following option creates an error-correction stream to a Helix Advanced Push broadcast by setting FECPercent to 100. The value of 1 for FECOffset delays the second stream by one second:

```
-sa TY45poi@helixserver.example.com:30001/sports/live.rm?FECPercent=100
&FECOffset=1
```

**For More Information:** Refer to the sections “Push Server Destinations” on page 75 and “Pull Server Destination” on page 83 for information about attributes used with push and pull broadcasts.

## Audiences

The audience settings determine the audio and video encoding parameters, such as the codecs and bit rates used. When you create a multi-rate clip, each audience defines the settings for one of the streaming rates. The following options are available.

-ad (audience file)	Selects an audience file.
-as (audience settings)	Specifies audio and video settings directly.

### Audience Options

The following command-line options affect audience settings for an output.

#### -ad (audience file)

Specifies audio and video encoding settings using an audience file.

value:	audience file name
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

#### Audience Templates

Audience templates are audience files stored in the audiences directory. You specify a template by using the file name or the value of the file's Name property (see page 55). You do not need to include the file path, and you can abbreviate a name to a unique set of characters:

```
-ad "65k RV"
```

**Tip:** Use the -pa (print audiences) option (see page 139) to display the available audience templates.

**For More Information:** See “Predefined Audience Templates” on page 143 for a list of default templates. The preferences file sets the audience template directory. See “File Path Preferences” on page 155.

#### Audience File Paths

If an audience file does not reside in the default audience directory, you can provide the full or relative path along with the file name and extension (.rpad). For example:

```
-ad "C:\settings\MyAudience.rpad"
-ad "../files/65k RV.rpad,/files/80k RV.rpad"
```

### Multiple Audiences for Multi-Rate CBR Encoding

When you use CBR encoding, you can specify multiple audiences to create a multi-rate RealMedia stream. There are two ways to do this:

- Separate each audience name with a comma and enclose the entire list in double quotes. For example:

```
-ad "65k RV,80k RV"
```

- Use the `-ad` option multiple times to specify each audience:

```
-ad "65k RV" -ad "80k RV"
```

**Note:** When using variable bit rate (VBR) encoding, you can specify only one audience.

### `-as` (audience settings)

Specifies audio and video settings on the command line.

value:	<i>AudioCodec:AvgBitrate[/MaxBitrate]#Channels ~VideoCodec:AvgBitrate[/MaxBitrate][@FrameRate][#WidthxHeight]</i>
default:	none
scope:	all outputs (outside braces) or individual outputs (inside braces)

**Note:** The `-as` option can define only one audience (a single-rate stream) for each output.

### Option Values

The following are the possible values. The Codec and AvgBitrate values are required.

<i>AudioCodec</i>	Name of the audio codec, such as ra-voice. Required. For a list of acceptable names, refer to “CodecName” on page 57.
<i>VideoCodec</i>	Name of the video codec, such as rv10. Required. For a list of acceptable names, refer to “CodecName” on page 60.
<i>AvgBitrate</i>	Average bit rate of the audio or video stream in Kilobits per second (Kbps). Required. A decimal point is allowed. Note that you can set audio codecs to specific bit rates only. Refer to <i>RealProducer User’s Guide</i> for information about the bit rates of all supported audio codecs.

<i>MaxBitrate</i>	Maximum bit rate of the audio or video stream in Kbps. Optional. Use an integer value, such as 160, to select VBRBitrate encoding. Preface the value with a Q, as in Q160, to choose VBRQuality encoding. If omitted, CBR encoding is used. For details about VBR encoding options, refer to “Rate Control Method” on page 65.
<i>Channels</i>	Channels to encode. Required. Options are: 1 – mono 2 – stereo 2s – stereo surround 5.1 – multichannel
<i>FrameRate</i>	Maximum frame rate in frames per second (fps). Optional. You can include a decimal point, as in 12.5.
<i>Width</i>	Width of the output video. Optional. If the width value is not a multiple of 4, the next lower multiple is used. For example, a value of 162 results in a video 160 pixels wide. If you set Width to 0 and Height to a specific value, the Width value scales automatically to a value that preserves the aspect ratio of the input. Fast resizing is always used.
<i>Height</i>	Height of the output video. Optional. If the value is not a multiple of 4, the next lower multiple is used. For example, a value of 127 results in a video 124 pixels high. If you set Height to 0 and Width to a specific value, the Height value scales automatically to a value that preserves the aspect ratio of the input. Fast resizing is always used.

#### Additional Parameters

You can specify additional parameters as query string parameters appended to the `-as` value using “?” and “&” operators. The following example encodes video at 160 Kbps with a maximum frame rate of 15 fps. It then sets the maximum keyframe interval to six seconds and the maximum buffering to three seconds:

```
-as AudioSettings~rv10:160@15?MaxKeyFrameInterval=6&MaxVideoBufferTime=3
```

**For More Information:** Refer to “Video Stream Properties” on page 59 for information about additional video properties.

## Audience Examples

The following examples show how to set audience encoding parameters for multiple outputs.

### Multiple Outputs with Audience Files

The following example uses audience templates to create a streaming clip at 100 Kbps and a download clip at a higher bandwidth:

```
producer -i input.avi
  {-o stream.rm -ad "100k RV" -ot rm}
  {-o download.rmvb -ad "750k RV" -ot rm}
```

### Multiple Outputs without Audience Files

The following example produces the same output as the preceding example using `-as` options to set the audio and video encoding parameters:

```
producer -i input.avi
  {-o stream.rm -as ra:20#2~rv10:80@15 -ot rm}
  {-o download.rmvb -as ra:64#2~rv10:686/788@30 -ot rm}
```

### Multiple Outputs Mixing `-ad` and `-as` Options

You cannot mix the use of `-ad` and `-as` in the same output. When you encode multiple outputs, however, you might specify an audience for one output using `-ad` and set the encoding parameters for another output using `-as`. The following example encodes two RealVideo outputs:

```
producer -i input.avi
  {-o output1.rm -ad "65k RV" -ot rm}
  {-o output2.rm -as ra:8#1~rv9:37@12.5 -ot rm}
```

The first output uses the encoding settings defined in the 65 Kbps RealVideo audience template. The second output is encoded using a RealAudio mono music codec at 8 Kbps. The video is encoded using RealVideo 9 at 37 Kbps and a maximum frame rate of 12.5 fps.

### Audio-Only Clip

To encode only the audio portion of an input, you can use an audience template that defines only the audio encoding parameters. To specify the encoding parameters on the command line, use the `-as` option, terminating the value string at the end of the audio portion.

The following example encodes only the audio portion of the input using a stereo RealAudio codec at 64 Kbps:

```
producer -i input.avi -o audio.rm -as ra:64#2 -ot rm
```

### Video-Only Clip

To encode only the video portion of an input, you can use an audience template that defines only the video encoding parameters. To specify the parameters on the command line, use the `-as` option, starting the value string at the tilde (`~`) delimiter that separates the audio and video strings.

The following example encodes only the video portion of the input using the RealVideo 10 codec at 128 Kbps:

```
producer -i input.avi -o video.rm -as ~rv10:128 -ot rm
```

## GENERAL OPTIONS

This chapter covers general command-line options that affect logging and display online help.

### Logging Options

Logging options affect the messages that RealProducer creates as it encodes a job.

-daw (disable audio watchdogs)	Disables audio tests.
-dlf (disable logging to file)	Disables logging to the standard log file.
-dls (disable logging to screen)	Disables logging to the screen.
-dvw (disable video watchdogs)	Disables video tests.
-lc (logging category)	Determines the types of messages logged.
-pid (process ID file)	Creates a process ID file.
-q (quiet mode)	Displays no information on the screen.

**For More Information:** The log file location is set through the RealProducer preferences. See “Log File Preferences” on page 160.

#### -daw (disable audio watchdogs)

Disables the audio watchdogs if added to the command line.

value:	(none)
default:	use audio watchdogs
scope:	all outputs (outside braces)

#### What are Audio Watchdogs?

Audio watchdogs are a set of tests that RealProducer runs on the audio input to detect audio problems. You can use these tests when encoding from a file or

a live capture, but they are most useful when capturing live media because you may be able to correct the problem using the capture equipment.

#### Audio Problem Logging

Audio problems are logged periodically (typically every five seconds) while the problem persists. After logging 20 messages for the same problem, RealProducer suspends logging of that message for 16 hours.

**For More Information:** Refer to the section “Audio Signal Detection Preferences” on page 168 for information about the audio tests performed when watchdogs are enabled. That section also explains the preference settings that affect the tolerance levels used by the watchdogs.

#### -dlf (disable logging to file)

Disables logging to the log file.

value:	no value required
default:	automatic logging to file
scope:	all outputs (outside braces)

**Tip:** Messages may still display on the screen.

#### -dls (disable logging to screen)

Disables logging to the screen.

value:	no value required
default:	automatic logging to screen
scope:	all outputs (outside braces)

**Tip:** Log messages may still be written to the log file.

**For More Information:** See “Screen Logging Preferences” on page 163.

#### -dvw (disable video watchdogs)

Disables the video watchdogs if added to the command line.

value:	(none)
default:	use video watchdogs
scope:	all outputs (outside braces)

### What are Video Watchdogs?

Video watchdogs are a set of tests that RealProducer runs on the video input to detect problems such as the lack of a signal. You can use these tests when encoding from a file or a live capture, but they are most useful when capturing live media because you may be able to correct the problem through the capture equipment.

### Video Problem Logging

Video problems are logged periodically (typically every five seconds) while the problem persists. After logging 20 messages for the same problem, RealProducer suspends logging of that message for 16 hours.

**For More Information:** Refer to the section “Video Signal Detection Preferences” on page 165 for information about the video tests performed when watchdogs are enabled. That section also explains the preference settings that affect the tolerance levels used by the watchdogs.

### -lc (logging category)

Determines the level of logging messages used.

value:	error warning informational diagnostic
default:	logging category set in preferences
scope:	all outputs (outside braces)

**Note:** These messages are written to the log file or printed to the screen, depending on the other logging options you choose.

### Logging Categories

For the logging category value, use a comma-separated string that includes any of the following:

error	Error messages indicate significant problems that have occurred in the encoding process.
warning	Warning messages indicate possible problems that may occur during the encoding.
informational	Informational messages provide important information about the encoding operation.
diagnostic	Diagnostic messages convey non-critical messages about the encoding operation.

**Note:** Because an encoding operation may generate many diagnostic messages, the log file can grow rapidly when you include the diagnostic category.

#### Examples

You can abbreviate each value down to the first letter, as shown in the following examples:

```
-lc "error,warning,info"  
-lc "err,warn,info,diag"  
-lc "e,w,i,d"
```

#### -pid (process ID file)

Writes a file that holds the process ID of the current encoding session.

value:	file name and path
default:	no PID file written
scope:	all outputs (outside braces)

**Tip:** This ID is useful for shutting down the command-line application remotely, as described in “Stopping the Application” on page 92.

#### PID File Path

The path can be absolute or relative to the working directory. Examples:

```
-pid ..\Jobs\job18.pid  
-pid /usr/log/job34.pid
```

#### -q (quiet mode)

Prevents any information, including log messages, from being printed to the screen.

value:	no value required
default:	messages logged to screen
scope:	all outputs (outside braces)

**Tip:** Use this option if you are running the command-line application from another application that will fail if output is written to the standard output device.

## Help Options

Help options display various types of information about the RealProducer command-line application.

-h (display help)	Displays available commands.
-m (display detailed help)	Provides full help for command line operation.
-pa (print audiences)	Prints audience definitions.
-pd (print device information)	Displays device information.
-ps (print servers)	Prints server definitions.
-v (print version)	Displays the version number.

**Note:** Do not include any other options on the command line when you specify a help option

### -h (display help)

Display a list of commands that you can use with the command-line application.

### -m (display detailed help)

Display full information about the commands that you can use with the command-line application.

### -pa (print audiences)

Prints a list of acceptable audiences.

**For More Information:** See -ad (audience file) option (see page 130).

### -pd (print device information)

Extracts audio and video capture device information. You can use the returned device information with the following options:

- -ac (audio capture device ID) option (see page 104)
- -vc (video capture device ID) option (see page 106)
- -vp (video device port) option (see page 107)

### -ps (print servers)

Lists the server definitions stored in the server templates directory.

**For More Information:** See “-sd (server definition file)” on page 125.

**-v (print version)**

---

Displays RealProducer version and copyright information.



## APPENDIXES

The following appendixes contain useful reference information.



## AUDIENCE TEMPLATES

Information in this appendix will help you to choose which audiences to use when encoding a clip or broadcast.

### Predefined Audience Templates

An audience template defines a range of parameters used to encode a clip or broadcast. For example, it defines the audio and video codec, along with the streaming bandwidth. RealProducer provides a number of predefined templates stored as files (file extension .rpad) in its audiences subdirectory.

### Templates Included with RealProducer

Most predefined video templates are available in three versions: RealVideo, H.263, and H.264. Audio-only templates are available for RealAudio, AAC, AAC+, and MP3. Each version uses the codecs and features appropriate for that output type.

**Note:** H.264, H.263, AAC, AAC+, and MP3 templates do not function with RealProducer.

#### Audio-Only Templates

The audio-only templates are for CBR audio output. They do not define any video settings:

- 16k Voice
- 32k Music
- 64k Music
- 96k Music

**Tip:** You can also use a video template to encode an audio-only output. RealProducer simply ignores the video settings.

## Video Templates (CBR)

Constant bit rate video templates define settings for streaming video and audio:

- 28k Video
- 56k Video
- 65k Video
- 80k Video
- 100k Video
- 150k Video
- 300k Video
- 750k Video
- 1.5M Video

**Tip:** You can add multiple CBR audiences to the same output to create a rate-shifting RealMedia stream.

## VBR Quality Template

The variable bit rate template is designed for downloadable video clips in the RealMedia format:

- 90% VBR Quality Video

**Note:** A VBR clip can encode only one audience.

## Customized Templates

RealProducer offers many more encoding choices than those available using the predefined templates. In many cases, you'll want to create your audience templates that cover your specific encoding needs. To do this, start with the predefined template that is closest to your desired output. Then edit the template using RealProducer. When it has the new settings you want, save the template under a new name.

## 16k Voice

The 16 Kb audio audiences target low-bandwidth, audio-only streaming of voice content (no music). There are three predefined audiences:

- 16k R5 Voice.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 16k R6 Voice.rpad—3GPP Release 6 or MPEG-4
- 16k RA Voice.rpad—RealAudio

The following table lists the encoding settings for these audiences.

<b>16k Audio Audience Settings</b>			
Setting	16k R5	16k R6	16k RA
Audio bandwidth	12.2 Kbps	15.85 Kbps	16 Kbps
Audio codec	AMR-NB	AMR-WB	RealAudio Voice
Audio channels	1	1	1
Rate control type	CBR	CBR	CBR

## 28k Video

The 28 Kb video audiences target low-bandwidth video streaming. There are three predefined audiences:

- 28k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 28k H.264.rpad—3GPP Release 6 or MPEG-4
- 28k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

<b>28k Video Audience Settings</b>			
Setting	28k H.263	28k H.264	28k RV
Video average bandwidth	20 Kbps	20 Kbps	20 Kbps
Audio bandwidth	8 Kbps	8 Kbps	6 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	1	1	1
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	7.5	7.5	7.5
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	20	20	20

**28k Video Audience Settings**

Setting	28k H.263	28k H.264	28k RV
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

**32k Music**

The 32 Kb audio audiences target low-bandwidth, audio-only streaming of music. There are four predefined audiences:

- 32k MP3 Music.rpad—MP3
- 32k AAC Music.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 32k AAC+ Music.rpad—3GPP Release 6 or MPEG-4
- 32k RA Music.rpad—RealAudio

The following table lists the encoding settings for these audiences.

**32k Audio Audience Settings**

Setting	32k MP3	32k AAC	32k AAC+	32k RA
Audio bandwidth	32 Kbps	32 Kbps	32 Kbps	32 Kbps
Audio codec	MP3	AAC-LC	AAC+	RealAudio
Audio channels	2	2	2	2
Rate control type	CBR	CBR	CBR	CBR

**56k Video**

The 56 Kb video audiences target low-bandwidth video streaming. There are three predefined audiences:

- 56k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 56k H.264.rpad—3GPP Release 6 or MPEG-4
- 56k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

#### 56k Video Audience Settings

Setting	56k H.263	56k H.264	56k RV
Video average bandwidth	44 Kbps	44 Kbps	44 Kbps
Audio bandwidth	12 Kbps	12 Kbps	12 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	1	1	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	7.5	7.5	7.5
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	30	30	30
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

## 64k Music

The 64 Kb audio audiences target low-bandwidth, audio-only streaming of music. There are four predefined audiences:

- 64k MP3 Music.rpad—MP3
- 64k AAC Music.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 64k AAC+ Music.rpad—3GPP Release 6 or MPEG-4
- 64k RA Music.rpad—RealAudio

The following table lists the encoding settings for these audiences.

#### 64k Audio Audience Settings

Setting	64k MP3	64k AAC	64k AAC+	64k RA
Audio bandwidth	64 Kbps	64 Kbps	64 Kbps	64 Kbps
Audio codec	MP3	AAC-LC	AAC+	RealAudio
Audio channels	2	2	2	2
Rate control type	CBR	CBR	CBR	CBR

## 65k Video

The 65 Kb video audiences target low-bandwidth video streaming. There are three predefined audiences:

- 65k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 65k H.264.rpad—3GPP Release 6 or MPEG-4
- 65k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

**65k Video Audience Settings**

Setting	65k H.263	65k H.264	65k RV
Video average bandwidth	53 Kbps	53 Kbps	53 Kbps
Audio bandwidth	12 Kbps	12 Kbps	12 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	1	1	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	10	10	10
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	35	35	35
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

## 80k Video

The 80 Kb video audiences target medium-bandwidth video streaming. There are three predefined audiences:

- 80k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 80k H.264.rpad—3GPP Release 6 or MPEG-4
- 80k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

#### 80k Video Audience Settings

Setting	80k H.263	80k H.264	80k RV
Video average bandwidth	64 Kbps	64 Kbps	64 Kbps
Audio bandwidth	16 Kbps	16 Kbps	16 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	1	1	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	15	15	15
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	40	40	40
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

## 96k Music

The 96 Kb audio audiences target high-bandwidth, audio-only streaming of music. There are four predefined audiences:

- 96k MP3 Music.rpad—MP3
- 96k AAC Music.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 96k AAC+ Music.rpad—3GPP Release 6 or MPEG-4
- 96k RA Music.rpad—RealAudio

The following table lists the encoding settings for these audiences.

#### 96k Audio Audience Settings

Setting	96k MP3	96k AAC	96k AAC+	96k RA
Audio bandwidth	96 Kbps	96 Kbps	96 Kbps	96 Kbps
Audio codec	MP3	AAC-LC	AAC+	RealAudio
Audio channels	2	2	2	2
Rate control type	CBR	CBR	CBR	CBR

## 100k Video

The 100 Kb video audiences target high-bandwidth video streaming. There are three predefined audiences:

- 100k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 100k H.264.rpad—3GPP Release 6 or MPEG-4
- 100k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

<b>100k Video Audience Settings</b>			
Setting	100k H.263	100k H.264	100k RV
Video average bandwidth	80 Kbps	80 Kbps	80 Kbps
Audio bandwidth	20 Kbps	20 Kbps	20 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	1	2	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	15	15	15
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	45	45	45
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

## 150k Video

The 150 Kb video audiences target high-bandwidth video streaming. There are three predefined audiences:

- 150k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 150k H.264.rpad—3GPP Release 6 or MPEG-4
- 150k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

#### 150k Video Audience Settings

Setting	150k H.263	150k H.264	150k RV
Video average bandwidth	122 Kbps	122 Kbps	118 Kbps
Audio bandwidth	28 Kbps	28 Kbps	32 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	2	2	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA
Video level	auto	auto	NA
Maximum video frame rate (fps)	15	15	15
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	50	50	50
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

## 300k Video

The 300 Kb video audiences target high-bandwidth video streaming. There are three predefined audiences:

- 300k H.263.rpad—3GPP Release 5, 3GPP Release 6, or MPEG-4
- 300k H.264.rpad—3GPP Release 6 or MPEG-4
- 300k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

#### 300k Video Audience Settings

Setting	300k H.263	300k H.264	300k RV
Video average bandwidth	244 Kbps	252 Kbps	256 Kbps
Audio bandwidth	56 Kbps	48 Kbps	44 Kbps
Audio codec	AAC-LC	AAC+	RealAudio
Audio channels	2	2	2
Video codec	H.263	H.264	RealVideo 10
Video profile	0	Baseline	NA

**300k Video Audience Settings**

Setting	300k H.263	300k H.264	300k RV
Video level	auto	auto	NA
Maximum video frame rate (fps)	30	30	30
Preroll (buffering time)	1 second	1 second	4 seconds
Quality	60	60	60
Encoding complexity	high	high	high
Rate control type	CBR	CBR	CBR

**750k Video**

The 750 Kb video audience targets medium-quality video downloading. There are two predefined audiences:

- 750k H.264.rpad—3GPP Release 6 or MPEG-4
- 750k RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

**750k Video Audience Settings**

Setting	750k H.264	750k RV
Maximum bandwidth	686 Kbps	686 Kbps
Video average bandwidth	686 Kbps	686 Kbps
Audio bandwidth	48 Kbps	64 Kbps
Audio codec	AAC+	RealAudio
Audio channels	2	2
Video codec	H.264	RealVideo 10
Video profile	Baseline	NA
Video level	auto	NA
Maximum video frame rate (fps)	30	30
Preroll (buffering time)	1 second	4 seconds
Quality	70	70
Encoding complexity	high	high
Rate control type	CBR	VBRBitrate

## 1.5M Video

The 1.5 Mb video audience targets high-quality video downloading. There are two predefined audiences:

- 1.5M H.264.rpad—3GPP Release 6 or MPEG-4
- 1.5M RV.rpad—RealVideo

The following table lists the encoding settings for these audiences.

**1.5M Video Audience Settings**

Setting	1.5M H.264	1.5M RV
Maximum bandwidth	1.404 Mbps	1.404 Mbps
Video average bandwidth	1.404 Mbps	1.404 Mbps
Audio bandwidth	96 Kbps	96 Kbps
Audio codec	AAC+	RealAudio
Audio channels	2	2
Video codec	H.264	RealVideo 10
Video profile	Baseline	NA
Video level	auto	NA
Maximum video frame rate (fps)	30	30
Preroll (buffering time)	1 second	4 seconds
Quality	80	80
Encoding complexity	high	high
Rate control type	CBR	VBRBitrate

## 90% VBR Quality Video

The 90% VBR Quality video audience targets high-quality video downloading. It attempts to preserve near-perfect video quality within the constraints of maximum bandwidth, frame size, and video content. There is one predefined audience:

- 90% VBR Quality RV.rpad—RealVideo

The following table lists the encoding settings for this audience.

**1.5M Video Audience Settings**

Setting	90% RV
Maximum bandwidth	2 Mbps
Video average bandwidth	2 Mbps
Audio bandwidth	96 Kbps
Audio codec	RealAudio
Audio channels	2
Video codec	RealVideo 10
Video profile	NA
Video level	NA
Maximum video frame rate (fps)	30
Preroll (buffering time)	4 seconds
Quality	90
Encoding complexity	high
Rate control type	VBRQuality

## PREFERENCES

The RealProducer preferences affect the operation of both the graphical application and the command-line application. You can edit this file manually if you need to change the preferences settings.

### Preferences File

RealProducer records application preferences in the file `producer.pref`, located in the main installation directory. This file, which you can modify using any text or XML editor, specifies values such as the paths to audience and server files. It also sets the location of the temporary directory used when encoding clips.

**Note:** This document does not describe the `applicationState` and `settingsFiles` properties, which are internal to RealProducer operation. You do not need to edit these properties.

### File Path Preferences

The `<paths/>` element sets the paths to files used by RealProducer. The default values set by the installation program typically do not need to be modified.

#### File Path Examples

The following example illustrates the `<paths/>` element:

```
<paths audiences=".\\audiences\\" servers=".\\servers\\" tempDir="%TEMP%"  
  licenseDir=".\\licenses\\"/>
```

#### File Path Attributes

The following attributes set the file path values.

### audiences

---

Provides the path to the directory that stores audience files.

value:	full path path relative to the main installation directory
default:	audiences subdirectory of the main installation directory

### licenseDir

---

Specifies the path to the directory that stores the license file.

value:	full path path relative to the main installation directory
default:	licenses subdirectory of the main installation directory

### servers

---

Specifies the path to the directory that stores server destination files.

value:	full path path relative to the main installation directory
default:	servers subdirectory of the main installation directory

### tempDir

---

Indicates the directory used to store temporary files during encoding.

value:	full path %TEMP% %OUTPUTDIR%
default:	%TEMP%

#### Directory Values

For tempDir, you can enter the full path name of the directory to use as the temporary directory. Or, you can use one of the following variables:

- %TEMP%

RealProducer uses the same directory specified by the Windows TEMP variable.

- %OUTPUTDIR%

Sets the temporary directory for each job to the same directory used to store the job's output clip.

**Tip:** RealProducer operates faster if the temporary directory resides on the local RealProducer machine, rather than a network drive.

## Output Preferences

The output preferences create default encoding settings such as the audience template used for an output type. Setting these preferences is useful if you repeatedly encode the same clip or broadcast, such as a program updated every hour. When you create a new job, RealProducer automatically fills in the appropriate fields with the default values.

**Note:** The default settings are used only with the graphical application. They do not apply to jobs processed by the command-line application.

### Output Settings Example

The following example illustrates the <defaults/> elements:

```
<defaults
  rmAudiences="80k RV,100k RV" r5Audiences="80k H.263"
  r6Audiences="80k H.264,100k H.264" mp3Audiences="96k MP3 Music"
  rmServers="" r5Servers="" r6Servers=""
  rmOutputFile="%InputFileDir%\%OutputName%.%OutputTypeExt%"
  r5OutputFile="%InputFileDir%\%OutputName%.%OutputTypeExt%"
  r6OutputFile="%InputFileDir%\%OutputName%.%OutputTypeExt%"
  mp3OutputFile="%InputFileDir%\%OutputName%.%OutputTypeExt%"
  rmTitle="" rmAuthor="RealNetworks, Inc." rmCopyright="(c) RealNetworks, Inc."
  rmKeywords="" rmRating="" rmRatingDescription=""
  r5Title="" r5Author="RealNetworks, Inc." r5Copyright="(c) RealNetworks, Inc."
  r5Keywords="" r5RatingDescription=""
  r6Title="" r6Author="RealNetworks, Inc." r6Copyright="(c) RealNetworks, Inc."
  r6Keywords="" r6Rating="" r6RatingEntity="" r6RatingDescription=""
  mp3Title="" mp3Author="RealNetworks, Inc." >
</defaults>
```

### Output Attribute Prefixes

Each output preference attribute includes a prefix that designates an output type:

mp3	MP3 audio
r5	3GPP Release 5
r6	3GPP Release 6
rm	RealMedia

For example, the `rmAudiences` attribute sets the default audiences for RealMedia outputs. The `r6Audiences` attribute sets the default audiences for 3GPP Release 6 outputs.

**Note:** Only the attributes with an `rm` prefix affect RealProducer.

## Output Attributes

The following are the output attributes organized alphabetically.

### xAudiences

Defines the default audience or audiences for the output type.

value:	audience template name  comma-separated list of audience template names
default:	none

**For More Information:** Refer to the section “Predefined Audience Templates” on page 143 for more about audiences.

### xAuthor

Indicates the author designation to include in the output stream.

value:	user-defined string
default:	none

### xCopyright

Provides a copyright string to include in the output stream.

value:	user-defined string
default:	none

### xKeywords

Encodes keywords into the output stream for search engines.

value:	list of keywords
default:	none

### xOutputFile

---

Sets the default output file name and location for the output type.

value:	full path and file name   path including variables
default:	none

#### File Name and Path Variables

For the output file name and path, you can use any of the following variables. RealProducer then substitutes the appropriate value when it runs the job.

%JobName%	Name of the current job file.
%InputFilename%	The input file name without the path or file extension.
%InputFileDir%	The input file's absolute directory path without the file name or extension.
%OutputName%	The output name.
%OutputType%	An output type designation: rm.
%OutputTypeExt%	The default file extension for the output type: rm.

### xRating

---

Sets a viewership rating.

value:	string
default:	none

**For More Information:** See “Metadata Values for RealMedia” on page 16.

### xRatingDescription

---

Explains the reason for the rating.

value:	string
default:	none

**For More Information:** See “Metadata Values for RealMedia” on page 16.

## xServers

---

Defines the default broadcast server for the output type.

value:	server template name
default:	none

**For More Information:** The section “-ps (print servers)” on page 139 explains how to display server template names.

## xTitle

---

Sets a title to include in the output stream.

value:	user-defined string
default:	none

## Log File Preferences

The `<fileLogging/>` element sets the RealProducer logging preferences. These preferences affect how both the command-line application and the graphical application log encoding events.

### File Logging Example

The following example illustrates the `<fileLogging/>` element:

```

<fileLogging
  format="detailed"
  formatSeparator="\t"
  disable="false"
  filename="producer.log"
  previousFilename="producer.log"
  enableRolling="false"
  rollType="time"
  rollSize="5"
  rollTimeIntervalType="monthly"
  filterFunctionalArea="false"
  functionalArea="all"
  category="error,warning,informational"
/>

```

## File Logging Attributes

The following are the file logging attributes organized alphabetically.

### category

---

Sets the logging category. Choose any combination of categories, separating the category names with commas.

value:	error warning informational diagnostic
default:	error,warning,informational

**For More Information:** See “Logging Categories” on page 137.

### disable

---

Turns off logging when set to true.

value:	true false
default:	false

### enableRolling

---

Enables the creation of multiple log files (log rolling) when set to true.

value:	true false
default:	false

**For More Information:** See also rollSize, rollTimeIntervalType, and rollType.

### filename

---

Sets the log file name.

value:	file name with full or relative path
default:	producer.log

**Note:** Relative path designations are relative to the main installation directory.

### filterFunctionalArea

---

Filters the log according to functional area when set to true.

value:	true false
default:	false

### format

---

Determines the log format.

value:	detailed short
default:	detailed

Values mean the following:

- detailed – Log full information, such as the message category, functional area, time, and message number.
- short – Log only the job name and message.

### formatSeparator

---

Defines the character used to separate entries on a line within the log.

value:	<i>character</i>
default:	\t (tab)

### functionalArea

---

Defines which functional areas are logged.

value:	all list of areas, comma-separated
default:	all

### previousFilename

---

Indicates the file name of the last log created if log rolling is used.

value:	file name
default:	producer.log <i>n</i>

**Note:** Do not edit the previousFilename value. Modify the log file name by changing the filename value.

### Numeric Extensions

The file extension includes a numeric designation to indicate the order of the logs. For example, for a log named `producer.log`, the first rolled file is named `producer.log1`, the second is `producer.log2`, and so on.

#### rollSize

Sets the size of the log file in Megabytes when log files are rolled according to size (`enableRolling` and `rollType` preferences).

value:	1-99999
default:	5

#### rollTimeIntervalType

Determines how frequently the log file is rolled if time-based rolling is turned on (`enableRolling` and `rollType` preferences).

value:	hourly daily weekly monthly
default:	monthly

#### rollType

Determines if log files are rolled according to size or preset time if log rolling is enabled (`enableRolling` preference).

value:	size time
default:	time

## Screen Logging Preferences

The screen logging preferences affect the events that are logged to the screen during the encoding process.

**Note:** The section “Log File Preferences” on page 160 explains the preference settings for writing information to the log file.

### Screen Logging Example

The following example illustrates the `<screenLogging/>` element:

```
<screenLogging
  category="error,warning,info"
  disable="false"
  filterFunctionalArea="false"
  functionalArea="all"
/>
```

## Screen Logging Attributes

The following preference settings affect screen logging.

### category

Sets the logging category. Choose any combination of categories, separating category names with commas.

value:	error warning informational diagnostic
default:	error,warning,informational

**For More Information:** See “Logging Categories” on page 137.

### disable

Turns off screen logging when set to true.

value:	true false
default:	false

### filterFunctionalArea

Filters log viewing according to functional area when set to true.

value:	true false
default:	false

**Note:** The default value of false displays all logging areas.

### functionalArea

Defines which functional areas are displayed.

value:	all list of functional areas, comma-separated
default:	all

**Tip:** The default value all displays all of the functional areas. You can also create a comma-separated list of specific areas.

## Video Signal Detection Preferences

These preferences affect video signal detection tests, which run automatically for all video streams. These tests write warnings to the log file if video data is missing. This is useful with live encoding to indicate problems with capture cards, cabling, cameras, and so on. The following example sets threshold values for the tests:

```
<VideoSignalDetectorPreFilter
  disable="false"
  FramePollInterval="30"
  NoSignalComplexityThreshold="5"
  NoSignalThresholdDuration="10"
  NoSignalReportingInterval="30"
  NoisySignalComplexityThreshold="95"
  NoisySignalThresholdDuration="10"
  NoisySignalReportingInterval="30"
  FrozenSignalComplexityChangeThreshold="2"
  FrozenSignalThresholdDuration="10"
  FrozenSignalReportingInterval="30"/>
```

### Video Conditions Detected

Using the preference settings, you can increase or decrease the sensitivity for the following video tests.

#### Frozen Video or Color Bars

The FrozenSignal values affect warnings that the video image has not changed. This condition can occur if the image is frozen or the video output displays color bars.

#### Random Noise

The NoisySignal values affect the tolerance for detecting white noise. This condition occurs when the video receiver cannot find a signal and reads background noise.

#### Single Color Image

The NoSignal values set the tolerances for detecting that the video image is a single, unchanging color. This typically indicates the lack of a video signal.

## Video Signal Detection Attributes

The following are the video signal detection properties organized alphabetically.

**Tip:** Typically, you can leave these values set to their defaults, changing them only if erroneous error messages are written to the log file.

### disable

---

Turns the video wathdog filter off if set to true.

value:	true false
default:	false

### FramePollInterval

---

Defines the interval at which the video signal is checked. This setting applies to all of the video tests.

value:	s[.xyz] up to 60.00
default:	30 (30 seconds)

### FrozenSignalComplexityChangeThreshold

---

Sets a factor used to gauge if video frames are changing.

value:	0-100 (0=solid image; 100=white noise)
default:	0.5

**Tip:** Lower the value if frozen signal warnings are consistently reported for video streams that are OK. Raise the value if frozen signals are underreported.

### FrozenSignalReportingInterval

---

Determines how frequently the frozen signal warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### FrozenSignalThresholdDuration

Defines the period over which the FrozenSignalComplexityChangeThreshold value is measured.

value:	[h:][m:]s[.xyz] up to 24:00:00
default:	10

### NoisySignalComplexityThreshold

Sets a complexity value used to identify if a video frame is white noise (background noise).

value:	0-100 (0=solid image; 100=white noise)
default:	95

**Tip:** Raise the value if white noise warnings are consistently reported for video streams that are OK. Lower the value if white noise conditions are underreported.

### NoisySignalReportingInterval

Determines how frequently the noisy signal warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### NoisySignalThresholdDuration

Defines how long a white noise signal can last before the condition is logged as an error.

value:	[h:][m:]s[.xyz] up to 24:00:00
default:	10

### NoSignalComplexityThreshold

Sets a complexity value used to identify if a video frame is a solid color, which indicates no signal.

value:	0-100 (0=solid image; 100=white noise)
default:	5

**Note:** This test does not identify color bars in the video stream. The complexity of color bars is significantly higher than a solid color image.

**Tip:** Lower the value if warnings about missing signals are consistently reported for video streams that are OK. Raise the value if missing signal conditions are underreported.

### NoSignalReportingInterval

Determines how frequently the no signal warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### NoSignalThresholdDuration

Defines how long a signal can go undetected before the condition is logged as an error.

value:	[h:][m:]s[.xyz] up to 24:00:00
default:	10

## Audio Signal Detection Preferences

These preferences affect audio signal detection tests, which run automatically for all audio streams. The tests write warnings to the log file if audio data is missing. This is useful with live audio to indicate problems with capture cards, cabling, microphones, and so on. The following example sets threshold values for the audio tests:

```
<AudioWatchdogPreFilter
  disable="false"
  SilenceThreshold="-60.00"
  SilenceThresholdDuration="10.00"
  SilenceReportingInterval="30.00"
  OutOfPhaseThresholdDuration="10.00"
  OutOfPhaseReportingInterval="30.00"
  ClippingThresholdDuration="10.00"
  ClippingReportingInterval="30.00"
  ChannelImbalanceThresholdDuration="10.00"
  ChannelImbalanceReportingInterval="30.00" />
```

## Audio Conditions Detected

Using the preference settings, you can increase or decrease the sensitivity for the following audio tests.

### Channel Imbalance

A channel imbalance occurs in stereo input when one channel is six decibels higher or lower than the other channel for a specified period. A warning indicates that the sound may be improperly weighted to one channel.

### Clipping

Clipping occurs when the audio peak reaches 0 decibels (maximum level) for four or more consecutive samples. At this point, distortion typically occurs. This test warns that the audio is being clipped heavily, moderately, or lightly.

### Out of Phase

This warning is logged if the left and right channels of a stereo signal are out of phase for more than 10 seconds.

### Silence

The silence test warns if the audio signal does not rise above a certain decibel setting. You can specify a period over which this condition must persist. This screens out normal audio lapses, such as when cutting to a commercial break.

## Audio Signal Detection Attributes

The following are the audio signal detection properties organized alphabetically.

**Tip:** Typically, you can leave these values set to their defaults, changing them only if erroneous error messages are written to the log file.

### ChannelImbalanceReportingInterval

Determines how frequently the channel imbalance warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### ChannellmbalanceThresholdDuration

Defines how long one channel of a stereo signal can be six dB higher or lower than the other channel before a warning is logged.

value:	s[.xyz] in the range 0 to 3600
default:	10

### ClippingReportingInterval

Determines how frequently the clipping warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### ClippingThresholdDuration

Defines a duration in seconds for which a clipped audio signal (0 dB) must exist before the condition is logged as a warning.

value:	s[.xyz] in the range 0 to 3600
default:	10

### disable

Turns the audio tests off if set to true.

value:	true false
default:	false

**Note:** This must be set before the encoding session begins.

### OutOfPhaseReportingInterval

Determines how frequently the out-of-phase warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### OutOfPhaseThresholdDuration

Defines a number of seconds for which the left and right channels of a stereo signal must be out of phase before the condition is logged as a warning.

value:	s[.xyz] in the range 0 to 3600
default:	10

### SilenceReportingInterval

Determines how frequently the silence warning is logged while the condition persists.

value:	[h:][m:]s[.xyz]
default:	30

### SilenceThreshold

Sets an audio level in dB at which the audio signal is considered to be silent.

value:	-s[.xyz] in the range -60.00 to 0
default:	-60

#### Audio Silence Levels

In digital audio signals, 0 is considered the maximum audio level. Each decrease of 6 decibels indicates a reduction in the audio level by half. A value of -60 represents a complete loss of the audio signal.

Set this property high enough to distinguish between background noise and true silence. In clean audio feeds, noise levels are low, and you can set this value low. For noisier feeds, such as those originating from analog sound cards, a low value may incorrectly identify noise as an active audio signal.

### SilenceThresholdDuration

Defines a duration in seconds for which silence must persist before the condition is logged as a warning.

value:	s[.xyz] in the range 0 to 3600
default:	10



## INDEX

- A** arch file name addition, 123
- audience file
  - audience section, 53
  - audio stream section, 56
  - broadcast latency, 54
  - compared to job file, 51
  - default preferences, 157
  - namespaces, 52
  - on command line, 130
  - overview, 51
  - packetizer, 67
    - interleaving duration, 70
    - packet size and duration, 69
    - properties, 68
    - superblocks, 70
  - rate control method
    - CBR, 65
    - VBRBitrate, 65
    - VBRQuality, 66
    - VBRUnconstrainedBitrate, 66
    - VBRUnconstrainedQuality, 66
  - video stream section, 59
- audience properties
  - AvgBitrate
    - for audio stream, 56
    - for video stream, 60
  - Channels, 57
  - CodecName
    - for audio, 57
    - for video, 60
  - EnableLossProtection, 61
  - EncodingComplexity
    - for audio, 58
    - for video, 61
  - FrameRateMode, 61
  - LatencyMode, 54
  - MaxBitrate, 62
  - MaxFramerate, 62
  - MaxKeyFrameInterval, 62
  - MaxPacketDuration, 55, 68
  - MaxPacketInterleavingDuration, 55, 68
  - MaxPacketSize, 55, 68
  - MaxVideoBufferTime, 63
  - OutputHeight, 63
  - OutputWidth, 63
  - PluginName
    - for audio, 58
    - for video, 64
  - Quality, 64
  - RateControlMethod
    - for video, 64
  - ResizeQuality, 65
- audience templates
  - 1.5M Video, 153
  - 100k Video, 150
  - 150k Video, 150
  - 16k Audio, 144
  - 28k Video, 145
  - 300k Video, 151
  - 32k Audio, 146
  - 56k Video, 146
  - 64k Audio, 147
  - 65k Video, 148
  - 750k Video, 152
  - 80k Video, 148
  - 90% Quality Video, 153
  - 96k Audio, 149
  - default location, 156
  - predefined files, 143
- audio delay compensation filter, 48
- audio gain filter
  - in job file, 47
  - on command line, 114
- audio noise detection

- ChannelImbalanceReportingInterval, 169
- ChannelImbalanceThresholdDuration, 170
- ClippingReportingInterval, 170
- ClippingThresholdDuration, 170
  - disable, 170
- OutOfPhaseReportingInterval, 170
- OutOfPhaseThresholdDuration, 171
- SilenceReportingInterval, 171
- SilenceThreshold, 171
- SilenceThresholdDuration, 171

audio stream

- average bit rate, 56
- channels, 57
- codec name, 57
- encoding complexity, 58
- examples, 56
- plug-in name, 58
- problem detection, 135
- properties, 56
- signal detection, 169

**B** batch encoding

- inputs, 108
- job file creation, 103
- output directory, 109

black-level correction

- in job file, 42
- on command line, 114

broadcasts

- Helix Advanced Push
  - in server file, 75
  - on command line, 124
- Helix Multicast Push
  - in server file, 75
  - on command line, 128
- Helix Pull
  - in server file, 83
  - on command line, 127
- Helix Push
  - in server file, 75
  - on command line, 128

latency

- in audience file, 54
- in job file, 28

- on command line, 120
- maximum packet size
  - in audience file, 69
  - in command line, 121
- see also* server destination file

**C** CBR, 65

- clip information, *see* metadata
- clipping detection, 169
- color bar detection, 165
- command-line application
  - audience options, 130
  - basics of using, 93
  - changes from earlier versions, 94
  - file destination options, 121
  - filter options, 113
  - help options, 139
  - input options, 103
  - job file
    - encoding overview, 96
    - options, 99
  - logging options, 135
  - metadata options, 110
  - output options, 120
  - overview, 91
  - return value
    - Window, 97
  - scope of options, 119
  - server destination options, 124
  - signal trapping, 91
  - signaling
    - Windows, 96
  - stopping, 91
  - syntax, 93
  - temporary directory, 92
  - usage examples
    - clip information, 112
    - file input, 108
    - job file, 102
    - live capture, 109
    - prefilters, 116
- command-line options
  - a author, 110
  - ac audio capture device ID, 104
  - ad audience file, 130

- as audience settings, 131
  - c copyright, 111
  - cj create job file, 99
  - cs capture frame size, 104
  - d capture duration, 105
  - daw audio watchdogs, 135
  - de description, 111
  - dlf disable logging to file, 136
  - dls disable logging to screen, 136
  - drs destination file roll size, 121
  - drt destination file roll time, 122
  - dt disable two-pass encoding, 122
  - dvw video watchdogs, 136
  - ej evaluate job file, 100
  - f:alg audio level gain filter, 114
  - f:vbl black-level filter, 114
  - f:vcr crop video, 114
  - f:vdI inverse-telecine and de-interlace, 115
  - f:vnf video noise filter, 115
  - f:vrs resize video, 116
  - h display help, 139
  - i input file or directory, 105
  - iep input end position, 106
  - isp input start position, 106
  - j job file name, 100
  - k keywords, 111
  - lc logging category, 137
  - lm latency mode, 120
  - m display detailed help, 139
  - mtu maximum packet size, 121
  - o output file or directory, 123
  - ot output type, 121
  - pa print audiences, 139
  - pd print device information, 139
  - pid process ID file, 138
  - ps print servers, 139
  - q quiet mode, 138
  - r content rating, 112
  - rp replace parameter, 101
  - sa Helix Advanced Push destination, 124
  - sd server template or file, 125
  - si Helix Pull destination, 127
  - sm Helix Multicast Push destination, 128
  - sp Helix Push destination, 128
  - t title, 112
  - v print version, 140
  - vc video capture device ID, 106
  - vp video device port, 107
- D**
- de-interlace filter
    - in job file, 39
    - on command line, 115
  - destinations
    - in job file, 25
    - on command line
      - file, 121
      - server, 124
- E**
- encoding complexity, 58
- F**
- file destinations
    - in job file, 30
    - on command line, 121
  - file rolling
    - in job file, 31
    - on command line, 122
- I**
- inputs
    - file
      - in job file, 17
      - on command line, 105
    - live capture
      - audio on command line, 104
      - in job file, 20
      - video on command line, 106
  - inverse-telecine filter
    - in job file, 39
    - on command line, 115
  - IP addresses, 127
- J**
- job file
    - archive files, 32
    - audience definitions, 51
    - audiences
      - importing from audience file, 52
      - section for defining, 51
    - audio
      - device ID and port, 24
      - filters
        - delay compensation, 48

- gain, 47
    - overview, 35
  - broadcast latency, 28
  - command line use, 99
  - destinations, 26
    - file, 30
    - server, 73
  - example, 10
  - file and path conventions, 18
  - file rolling, 31
  - inputs
    - capture, 20
    - examples
      - file input, 18
      - live capture, 20
    - file, 17
    - multiple, 17
    - single, 16
  - job section, 10
  - metadata, 13, 14
  - outputs
    - file name conventions, 32
    - output section, 28
  - overview, 7
  - parallel inputs
    - defining, 17
    - example, 21
  - updating syntax from earlier version, 102
  - variables for attribute values, 101
  - video
    - cropping, 37
    - device ID and port, 24
    - filters
      - black-level, 42
      - de-interlace, 39
      - inverse-telecine, 39
      - noise reduction, 41
      - order for using, 36
      - overview, 35
      - resizing, 43
    - resizing
      - width and height, 46
    - resizing methods, 44
  - job properties
    - AudioDeviceID, 21
    - AudioDevicePort, 21
    - Destination, 26
    - DestinationRollSize, 31
    - DestinationRollTime, 31
    - Destinations, 26
    - DisableLoadManagement, 11
    - DisableWallclock, 11
    - Duration, 22
    - EnableTwoPass, 11
    - EndPosition, 18
    - Filename, 18, 32
    - Input, 16
    - LatencyMode, 28
    - LoadManagementJitterThreshold, 11
    - LoadManagementLatencyMax, 12
    - LoadManagementLatencyTreshold, 12
    - MaxPacketDuration, 29
    - MaxPacketInterleavingDuration, 29
    - MaxPacketSize, 29
    - Metadata, 13
    - Name, 30, 33
      - for captures, 22
      - for inputs, 19
      - for metadata, 15
    - Output, 28
    - Outputs, 26
    - OutputType, 30
    - ParInputs, 16
    - ParOutputs, 26
    - PluginName, 19, 33
      - for captures, 22
    - StartPosition, 19
    - Type, 15
    - Value, 15
    - VideoDeviceID, 23
    - VideoDevicePort, 23
    - VideoFrameHeight, 23
    - VideoFramerate, 24
    - VideoFrameWidth, 24
- L**
- license
    - directory, 156
  - logging
    - category
      - on command line, 137
      - preference setting, 161
    - disabling

- on command line, 136
    - preference for, 161
  - file name, 161
  - file rolling
    - by size or time, 163
    - enabling, 161
    - file names, 162
    - file size, 163
    - interval, 163
  - format, 162
  - functional areas
    - defining, 162
    - filtering by, 162
  - line separator, 162
  - on screen
    - category, 164
    - disable, 164
    - filter functional area, 164
    - functional area, 164
- M** metadata
- attributes, 14
  - in job file, 13
  - on command line, 110
  - RealMedia, 16
- O** output preferences
- audiences, 158
  - author, 158
  - copyright, 158
  - keywords, 158
  - output file, 159
  - rating, 159
  - rating description, 159
  - servers, 160
  - title, 160
- output type
- in job file, 30
  - on command line, 121
- outputs
- in job file, 28
  - on command line, 120
- P** paths
- relative and absolute, 19
- pixel aspect ratio, 44
- preferences
- application state, 155
  - audio signal detection, 168
  - default audiences, 157
  - file paths, 155
  - log viewing, 163
  - logging, 160
  - output settings, 157
  - video signal detection, 165
- prefilter attributes
- audio delay compensation
    - Advance, 49
    - Delay, 49
    - Enabled, 49
    - PluginName, 50
    - xsi:type, 50
  - audio gain
    - Enabled, 47
    - Gain, 47
    - PluginName, 48
    - xsi:type, 48
  - black level
    - Enabled, 43
    - PluginName, 43
    - xsi:type, 43
  - deinterlace
    - Deinterlace, 40
    - Enabled, 40
    - Manual, 40
    - xsi:type, 41
  - input cropping
    - Enabled, 37
    - Height, 38
    - Left, 38
    - PluginName, 38
    - Top, 38
    - Width, 39
    - xsi:type, 39
  - inverse-telecine
    - Enabled, 40
    - InverseTelecine, 40
    - Manual, 40
    - xsi:type, 41
  - noise reduction
    - Enabled, 42

- Level, 42
- PluginName, 42
- xsi:type, 42
- video resize
  - Enabled, 45
  - PluginName, 45
  - VideoResizeHeight, 45
  - VideoResizeWidth, 46
  - xsi:type, 46
- prefilters on command line
  - f:alg audio level gain filter, 114
  - f:vbl black-level filter, 114
  - f:vcr crop video, 114
  - f:vdj, 115
  - f:vnf, 115
  - f:vrs, 116
- process ID, 138
  - Windows, 96
- R**
  - rate control method
    - CBR, 65
    - in video stream, 64
    - VBRBitrate, 65
    - VBRQuality, 66
    - VBRUnconstrainedBitrate, 66
    - VBRUnconstrainedQuality, 66
  - RealVideo
    - frame rate mode, 61
  - relative paths, 19
  - .rpad extension, 52
  - .rpjf extension, 7
  - .rpsd extension, 73
- S**
  - scope of command-line options, 119
  - server destination file
    - broadcast types, 75
    - examples
      - pull, 86
      - push, 82
    - on command line, 124
    - overview, 73
    - properties
      - pull methods, 84
      - push methods, 76
    - using as template, 74
  - server destinations
    - template location, 156
  - server properties
    - Destination, 74
  - pull broadcast method
    - EncoderAddress, 84
    - EncoderListenPort, 84
    - KeepAliveTimeout, 84
    - Name, 85
    - Password, 85
    - PluginName, 85
    - SavePassword, 86
    - Streamname, 86
  - push broadcast methods
    - AcceptResends, 76
    - Address, 77
    - FECOffset, 77
    - FECPercent, 77
    - HTTPPort, 77
    - MulticastAddress, 78
    - MulticastPort, 78
    - MulticastTTL, 78
    - Name, 78
    - Password, 79
    - PluginName, 79
    - Port, 79
    - ReconnectInterval, 80
    - ResendListenAddress, 80
    - SavePassword, 80
    - StatisticsUpdateInterval, 81
    - Streamname, 81
    - Transport, 81
    - Username, 82
  - Server, 74
  - signal producer
    - Windows, 97
  - signaling
    - Windows, 96
  - silence detection, 169
- T**
  - temporary directory, 156
  - two-pass encoding
    - in job file, 11
    - on command line, 122

- V**
  - VBRBitrate, 65
  - VBRQuality, 66
  - VBRUnconstrainedBitrate, 66
  - VBRUnconstrainedQuality, 66
  - video noise detection
    - FramePollInterval, 166
    - FrozenSignalComplexityChangeThreshold, 166
    - FrozenSignalReportingInterval, 166
    - FrozenSignalThresholdDuration, 167
    - NoisySignalComplexityThreshold, 167
    - NoisySignalReportingInterval, 167
    - NoisySignalThresholdDuration, 167
    - NoSignalComplexityThreshold, 167
    - NoSignalReportingInterval, 168
    - NoSignalThresholdDuration, 168
  - video stream, 165
    - bit rate
      - average, 60
      - maximum, 62
    - buffer time, 63
    - codec
      - name, 60
    - cropping
      - in job file, 37
      - on command line, 114
    - encoding complexity, 61
    - example in audience file, 59
    - filters
      - in job file, 35
      - on command line, 113
    - frame rate maximum, 62
    - keyframe interval, 62
    - loss protection, 61
    - noise reduction
      - in job file, 41
      - on command line, 115
    - plug-in name, 64
    - preroll, 63
    - problem detection, 136
    - properties, 59
    - quality, 64
    - rate control method, 64
    - resizing
      - input, 43
      - height, 45
      - quality, 46
      - width, 46
    - methods for resizing, 44
    - on command line
      - dimensions, 116
      - quality, 116
    - output
      - height, 63
      - quality, 65
      - width, 63
    - signal detection, 165
    - white noise detection, 165
- W**
  - watchdog tests
    - audio, 135
    - video, 136
  - white noise detection, 165

