



Wowza Streaming Engine™

User Guide

Wowza Streaming Engine User Guide



Version: 4.8

www.wowza.com

This document is for informational purposes only and in no way shall be interpreted or construed to create warranties of any kind, either express or implied, regarding the information contained herein.

No Endorsement or Warranty for Third-Party Links and Software

This document contains links to third-party websites ("Linked Sites") that are not under the control of Wowza Media Systems™, LLC ("Wowza"). Wowza is not responsible for the content on or operation of Linked Sites. If you access Linked Sites, you do so at your own risk and understand that Wowza accepts no responsibility or liability for the content or operation of Linked Sites. Wowza provides these links only as a convenience, and the inclusion of a link does not imply that Wowza endorses such Linked Sites or any content, products, or services available from Linked Sites.

This document also refers to third-party software that is not licensed, sold, or distributed by Wowza (collectively, "Third-Party Software"). Wowza does not endorse, is not responsible for, and accepts no liability related to Third-Party Software. Please ensure that any and all use of Wowza software and third-party software is properly licensed.

Wowza Trademarks

Wowza™, Wowza GoCoder™, Wowza™ Player, Wowza Streaming Cloud™, Wowza Streaming Engine™, and other words and phrases, along with other logos, trade dress, and other proprietary colors and markings, are trademarks or registered trademarks of Wowza in the United States and in other countries (collectively, "Wowza Marks"). No right to use Wowza Marks in any way is granted hereunder. Contact sales@wowza.com for information about obtaining the right to use Wowza Marks. Any use of Wowza Marks, authorized or otherwise, shall inure to the sole benefit of Wowza.

Third-Party Trademarks and Copyrights

Trademarks, product names, logos, designs, trade dress, and other proprietary markings of non-Wowza third parties (collectively, "Third-Party Marks") may be trademarks or registered trademarks of their respective owners. Use of Third-Party Marks is for the sole purpose of identifying third-party products and services and does not represent endorsement, sponsorship, partnership, or other affiliation between Wowza and such third parties.

A list of applicable patent and copyright notices related to content in this document is available on the Wowza website at www.wowza.com/legal.

Except as may be permitted by law, no part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Wowza Media Systems.

Document history

Version	Description	Date
Doc v4.8.0	Document release for Wowza Streaming Engine 4.8.0	02-05-2020

Note

More complete and up-to-date documentation is available online. See [Wowza Streaming Engine product articles](#) for the latest content.

Table of contents

What's new.....	7
Transcoder enhancements	7
Full support for HEVC/H.265 streaming.....	7
Improvements since Wowza Streaming Engine 4.7.0.....	7
Streaming protocols and playback	9
HLS.....	9
MPEG-DASH	10
WebRTC	11
SRT.....	12
HDS	12
Smooth Streaming	13
RTMP.....	13
RTSP/RTP	14
Supported container formats for HLS and MPEG-DASH streaming	15
Supported media file formats for VOD streaming	15
Additional protocol and format details	16
Live stream transcoding and transrating	16
Live stream nDVR.....	18
Stream encryption with DRM	18
AddOns.....	19
Installed examples	20
Server installation	21
Before you start	21
Installing Wowza Streaming Engine	22
Starting and stopping the software	23
Uninstalling Wowza Streaming Engine	27
Running Wowza Streaming Engine as a named user.....	27
Entering a license key.....	27
Ports used for streaming.....	30
Server configuration and tuning	31
Software updates	32
Application configuration	33
Applications and application instances	34

WOWZA STREAMING ENGINE 4.8 USER GUIDE

Playback URL formats	34
Stream types.....	36
HTTP streamers and live stream packetizers	37
Timed text providers	40
Transcoder and nDVR configurations	42
Modules	42
Properties	43
Media types	44
Content storage	46
Advanced configuration.....	48
MediaCaster, Stream files, and Startup Streams.....	48
Live stream repeater (origin/edge live streaming)	51
Live stream recording	54
Virtual hosting	55
Publishing with the Stream and Publisher classes.....	58
Using Wowza Streaming Engine Manger.....	59
Starting and stopping Wowza Streaming Engine Manager.....	59
Managing sign-in credentials	62
Navigating Wowza Streaming Engine Manager.....	64
Server administration	76
Configuring SSL and RTMPS	76
Logging	77
More resources	83

What's new

Wowza Streaming Engine™ media server software is a robust, customizable, and highly extensible platform that powers live and on-demand adaptive bitrate streaming to any device, anywhere. Wowza Streaming Engine 4.8.0 contains several fixes and enhancements that improve the functionality of the media server as well as some exciting new features.

Transcoder enhancements

Wowza Streaming Engine 4.8.0 includes a variety of improvements to the Transcoder feature. Wowza Transcoder now supports hardware accelerated decoding of HEVC/H.265 on NVIDIA NVCUVID-based GPUs. The libvpx library was also upgraded to version 1.8.1 to improve VP8 and VP9 transcoding.

Full support for HEVC/H.265 streaming

Previous versions of Wowza Streaming Engine supported HEVC/H.265 streaming as a preview feature. This workflow is now fully supported. However, it's important to note that HEVC/H.265 video compression hasn't been widely adopted by player vendors, which means your options will be limited for delivering these streams to viewers. For more information, see [Stream using HEVC/H.265 with Transcoder in Wowza Streaming Engine](#).

Improvements since Wowza Streaming Engine 4.7.0

Wowza Streaming Engine 4.8.0 contains all the big features and improvements since Wowza Streaming Engine 4.7.0. These include full support for [WebRTC](#) and [Secure Reliable Transport \(SRT\)](#) streaming; the addition of the [CMAF packetizer for MPEG-DASH, HLS, and Low-Latency HLS streaming](#); and added support for [recording MPEG-DASH live streams with the nDVR feature](#).

Additionally, Wowza Streaming Engine 4.8.0 is built on Java 9 (OpenJDK Java SE JRE 9.0.4) and can be used with Java versions 9-12. This enables you to use your Wowza Streaming Engine media server software with Java SE 11, which is a long-term supported version.

However, because of changes between Java versions, if your Wowza Streaming Engine workflow uses custom modules or plugins, we recommend that you test them in a non-production environment prior to updating your production media server software or Java version. For more information, see [Update to Java 9](#).

Streaming protocols and playback

Wowza Streaming Engine supports a wide variety of streaming protocols and formats to enable live and video-on-demand (VOD) streams to play on the desktop, mobile devices, set-top boxes, and more. Here's an overview of all of the protocols and formats that Wowza Streaming Engine supports to help you understand both what the protocols are, and which devices and players use each protocol.

For more information, see our technical article [Understanding protocols and formats supported by Wowza Streaming Engine](#).

HLS

Wowza Streaming Engine uses HTTP Live Streaming (HLS) to deliver adaptive bitrate live and VOD content to iOS devices (iOS 3.0 or later); Wowza Player and QuickTime Player (version 10 or later); Safari (4.0 or later); devices such as the Roku and Amino set-top boxes; and some smart TVs. HLS is a segment-based protocol that uses HTTP for delivery.

Wowza Streaming Engine delivers HLS streams in any of three ways.

First, using the "Cupertino" packetizer (**cupertinostreamingpacketizer**), the original HLS packetizer in Wowza Streaming Engine, the server software packages the content into keyframe-aligned segments that it calls *chunks*. These segments use the MPEG-TS container format. Cupertino streaming, as it's called, can be used for both live and VOD streaming. Cupertino streaming uses HTTP/1.1 and supports the following codecs:

- **Video** – H.264
- **Audio** – AAC, AAC-LC, and HE-AAC (AAC+ or aacPlus); Dolby® Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3); MP3

Alternatively, you can use the newer CMAF packetizer (**`cmfstreamingpacketizer`**) to package HLS live streams that contain keyframe-aligned media segments wrapped in the fMP4 container format instead of the MPEG-TS format. Like Cupertino streams, CMAF-packaged HLS streams are delivered to players over HTTP/1.1. However, in addition to using a different container format, CMAF-packaged HLS streams support slightly different codecs:

- **Video** – H.264, H.265
- **Audio** – AAC, AAC-LC, HE-AAC (AAC+ or aacPlus), HE-AACv2 (enhanced AAC+, aacPlus v2); Dolby Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3)

Finally, Wowza Streaming Engine can also produce Low-Latency HLS live streams, which can be played in native apps on devices running iOS 13. Wowza Streaming Engine generates Low-Latency HLS streams using the CMAF packetizer, which, in addition to generating CMAF segments, also generates smaller units of streaming media called *chunks* in the code base. These CMAF-compliant chunks also serve as *partial segments* for Low-Latency HLS streaming. As a result, you can use the CMAF packetizer to generate Low-Latency HLS streams whose segments (and partial segments) are wrapped in the fMP4 format and delivered to players over HTTP/2.

Wowza Streaming Engine populates the playlist file with metadata that describes each of the available streams in an adaptive bitrate presentation. This enables supporting players to select the appropriate streams based on hardware device capabilities.

Wowza Streaming Engine supports multiple encryption methods for protecting HLS streams using DRM. See [Secure HLS streaming using DRM encryption with Wowza Streaming Engine](#).

Wowza Streaming Engine can also send timed data events to the iOS player in the form of ID3 tags. See [Convert timed metadata from AMF to ID3 using the Wowza Streaming Engine Java API](#).

MPEG-DASH

Wowza Streaming Engine can deliver MPEG-DASH (Dynamic Adaptive Streaming over HTTP) adaptive bitrate live and VOD content to clients that can play MPEG-DASH streams. MPEG-DASH is an ISO standard ([ISO/IEC 23009-1](#)) for streaming segment-based content over HTTP.

Wowza Streaming Engine provides two packetizers that perform all of the segmenting and packaging necessary to deliver MPEG-DASH streams. The first, **mpegdashstreamingpacketizer**, packetizes the content into keyframe-aligned segments that it calls *chunks*. These segments use the fMP4 container format. MPEG-DASH streaming can be used for both live and VOD streaming and supports the following codecs:

- **Video** – H.264, H.265
- **Audio** – AAC, AAC-LC, HE-AAC (AAC+ or aacPlus), HE-AACv2 (enhanced AAC+, aacPlus v2); Dolby Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3); (Wowza Streaming Engine 4.7.2.01 or later) MPEG-4 Audio Lossless Coding (ALS)

The newer **cmfistreamingpacketizer** packetizer also segments and packages MPEG-DASH live streams. CMAF streaming packetizes content into keyframe-aligned segments that use the fMP4 container format. The CMAF packetizer supports the following codecs for live streaming over MPEG-DASH:

- **Video** – H.264, H.265
- **Audio** – AAC, AAC-LC, HE-AAC (AAC+ or aacPlus), HE-AACv2 (enhanced AAC+, aacPlus v2); Dolby Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3)

Wowza Streaming Engine provides MPEG-DASH players with a list of the available media segment URLs in a Media Presentation Description (MPD) manifest. The MPD describes segment information such as timing, language, timed text, and media characteristics (video resolution and bitrate). Players request media segments sequentially based on network conditions, device capabilities, and other factors to enable uninterrupted playback of the adaptive bitrate media presentation.

MPEG-DASH is codec-agnostic and supports multiplexed and non-multiplexed encoding. Multiple content protection (DRM) schemes are supported; Common Encryption (CENC) can also be used to encrypt MPEG-DASH streams once and deliver the single encrypted stream to players that support different licensing systems. For more information see [Secure MPEG-DASH streams using Common Encryption in Wowza Streaming Engine](#).

WebRTC

Wowza Streaming Engine can stream live and VOD content to browser players that support WebRTC streams. WebRTC is an open-source project designed to provide browsers and mobile applications with real-time communication capabilities.

Wowza Streaming Engine supports the following video and audio codecs when using this streaming protocol:

- **Video** – VP8, VP9, H.264
- **Audio** – Opus, Vorbis, Pulse Code Modulation (PCM) types PCMU and PCMA

For more information about playing WebRTC streams, see [Play WebRTC streams from Wowza Streaming Engine](#).

SRT

Wowza Streaming Engine supports MPEG-TS-based Secure Reliable Transport (SRT) in Linux and Windows server installations. A MediaCaster type enables live applications to ingest SRT source streams and make them available to all player technologies supported by Wowza Streaming Engine, and the generic SRT stream target enables you to deliver the SRT content directly to SRT destinations.

Wowza Streaming Engine supports these codecs for SRT:

- **Video** – H.264, H.265, VP8, VP9
- **Audio** – AAC, AAC-LC, HE-AAC (AAC+ or aacPlus), HE-AACv2 (enhanced AAC+, aacPlus v2); MP3, AC-3 (Dolby® Digital); E-AC-3 (Dolby Digital Plus); ALS (LOAS); Opus; Vorbis

For more information, see [Ingest and publish an SRT stream with Wowza Streaming Engine](#) and [Use SRT to distribute live streams from Wowza Streaming Engine](#).

HDS

Wowza Streaming Engine can stream adaptive bitrate live and VOD content to Adobe Flash Player 10.1 or later using the HTTP Dynamic Streaming (HDS) protocol. Developed by Adobe, HDS is a chunk-based streaming protocol that uses HTTP for delivery. All media-chunking and packaging necessary to deliver a stream using this protocol is performed by Wowza Streaming Engine. HDS is referred to as "San Jose" streaming in the Wowza Streaming Engine configuration files.

When streaming VOD content, Wowza Streaming Engine software supports MP4 files (QuickTime container) and MP3 files. FLV files are supported for RTMP playback. Wowza Streaming Engine supports the following video and audio codecs when using the HDS protocol:

- **Video** – H.264, On2 VP6 (live only), Screen video and Screen video 2 (live only), Sorenson Spark (live only)
- **Audio** – AAC, AAC Low Complexity (AAC LC), AAC High Efficiency (HE-AAC) v1 and v2; MP3; Speex (live only)

Smooth Streaming

Wowza Streaming Engine can stream adaptive bitrate live and VOD H.264, AAC, and MP3 content to Microsoft Silverlight, Windows Phone devices, and other devices using the Smooth Streaming protocol from Microsoft. Microsoft Silverlight is a cross-browser, cross-platform technology. Smooth Streaming is a chunk-based streaming protocol that uses HTTP for delivery. All media chunking and packaging necessary to deliver a stream using this protocol is performed by Wowza Streaming Engine, so there's no need for an IIS web server.

The following media formats can be used when streaming to Smooth Streaming clients:

- **Video** – H.264
- **Audio** – AAC, AAC Low Complexity (AAC LC), AAC High Efficiency (HE-AAC) v1 and v2; MP3

RTMP

Wowza Streaming Engine can send adaptive bitrate live and VOD streams to Adobe Flash Player using the Real Time Messaging Protocol (RTMP). Wowza Streaming Engine supports all video and audio formats that Flash Player supports:

- **Video** – H.264, On2 VP6, Sorenson Spark, Screen video and Screen video 2
- **Audio** – AAC, AAC Low Complexity (AAC LC), AAC High Efficiency (HE-AAC) v1 and v2, MP3, Speex

Wowza Streaming Engine supports the following RTMP protocol variants:

- **RTMP** – The base protocol and the most efficient and fastest of the variants.
- **RTMPE** – A lightweight encryption variant that helps to secure the data being transmitted between Wowza Streaming Engine and Flash Player.
- **RTMPS** – An encryption variant that transmits data over a secure SSL connection. RTMPS uses a more robust encryption layer than RTMPE to wrap the RTMP session. Users with Subscription and Perpetual licenses for Wowza Streaming Engine software can use [Wowza StreamLock™ AddOn](#) to get free 256-bit SSL certificates for use with RTMP, RTMPS, HTTP, and HTTPS.
- **RTMPT** – A tunneling variant that is used to tunnel through firewalls that employ stateful packet inspection.
- **RTMPTE** – An encryption variant of the RTMPT protocol.

Wowza Streaming Engine includes bi-directional support for Action Message Format (AMF3 and AMF0) for data serialization.

RTSP/RTP

Wowza Streaming Engine can stream live H.264, AAC, and MP3 content to players and devices that support the Real Time Streaming Protocol (RTSP), Real-time Transport Protocol (RTP), and MPEG-2 Transport Stream protocol (MPEG-2 TS). This includes players and devices such as QuickTime player (version 10 or later), VideoLAN VLC player, set-top boxes, and 3GPP devices. Wowza Streaming Engine also accepts incoming source streams from encoding devices that use these protocols, and supports RTP and MPEG-2 TS input and output over UDP as well as multicast. In addition, Wowza Streaming Engine supports interleaved RTSP/RTP (RTP over the RTSP TCP connection) and RTSP/RTP tunneling (RTSP/RTP over HTTP), which enables RTSP/RTP to be delivered in network environments that don't allow UDP transmission.

Wowza Streaming Engine supports the following RTSP, RTP, and MPEG specifications:

MPEG-TS	ISO/IEC 13818-1
MPEG-TS over RTP	rfc2038
RTP: AAC	rfc3640, rfc3016, ISO/IEC 14496-3
RTP: G.711	rfc3551
RTP: H.263	rfc2429
RTP: H.264	rfc3984, QuickTime Generic RTP Payload Format
RTP: MP3	rfc2250
RTP: MPEG-2 (video)	rfc2250
RTP: MPEG-4 Part 2	rfc3106
RTP: Speex	rfc5574
RTSP	rfc2326

Wowza Streaming Engine supports both Single Program (SPTS) and Multi Program (MPTS) MPEG-TS streams and enables you to specify a specific program, a specific language, and a specific audio or video track in an MPTS stream.

Query parameters are part of the udp:// URL in a .stream file. There are four options for selecting a stream. For more information, see [Specify per-stream settings in Wowza Streaming Engine .stream files](#).

Supported container formats for HLS and MPEG-DASH streaming

A *container format* is a wrapper for segments of content that are delivered over HTTP-based streaming protocols. Wowza Streaming Engine supports two container formats for HLS and MPEG-DASH live streaming: fragmented MP4 and MPEG-TS.

Fragmented MP4

The ISO standards MPEG-DASH and CMAF use the fragmented MP4 (fMP4) container format. The MPEG-DASH and CMAF packetizers in Wowza Streaming Engine (**mpegdashstreamingpacketizer** and **cmastreamingpacketizer**, respectively) wrap stream segments in the fMP4 container format. HLS master playlists and MPEG-DASH manifests generated by the CMAF packetizer reference these fMP4 files and make them available to players when they request available segments for playback.

MPEG-TS

MPEG Transport Stream (MPEG-TS) is the container format defined in the MPEG-2 ISO standard. The Cupertino packetizer (**cupertinostreamingpacketizer**) in Wowza Streaming Engine wraps stream segments (what it calls *chunks*) in the MPEG-TS container format. The HLS master playlist generated by the Cupertino packetizer references these MPEG-TS files and makes them available to players when they request available segments for playback.

Supported media file formats for VOD streaming

For VOD streaming, Wowza Streaming Engine supports the MP4 (QuickTime container), FLV (Flash Video), MP3, and SMIL (Synchronized Multimedia Integration Language) media file formats. To play VOD content, the correct prefix and extensions must be appended to the file name to create a *stream name*. For example, to play the MP4 file **mycoolvideo.mov**, use the stream name **mp4:mycoolvideo.mov**.

	Prefix	Extension(s)	Example
MP4	mp4:	.mp4, .f4v, .mov, .m4a, .m4v, .mp4a, .mp4v, .3gp, .3g2	mp4:mycoolvideo.mov
MP3	mp3:	.mp3	mp3:mycoolsong.mp3
SMIL	smil:	.smil	smil:myStream.smil
FLV	flv:	.flv	flv:mycoolvideo.flv

Note

MP4 (QuickTime container) is the default media type, so the file name prefix and extension can be omitted.

The media prefix also controls the file container that stores recorded live video. If mp4: or if no prefix is specified, the content is recorded to an MP4 (QuickTime) container. Only H.264, AAC, and MP3 content can be recorded to an MP4 container. If flv: is specified, an FLV (Flash Video) container is used.

Additional protocol and format details

Wowza Streaming Engine can accept live video and audio streams from sources that support the RTMP, RTSP/RTP, native RTP, and MPEG-TS protocols. It can record any live source stream to either the MP4 or FLV format.

Wowza Streaming Engine can read and write Action Message Format (AMF0 and AMF3) data events to and from MP4 files. In addition, it supports MP4 multi-language caption and audio tracks.

Wowza Streaming Engine can be used to re-stream SHOUTcast and Icecast (AAC, AAC+, and MP3) audio streams and IP camera (AAC, G.711 (μ-law and A-law), H.264, and MP3) streams to supported player technologies. It maintains a single connection to the original source stream while delivering the stream to multiple players. It can also forward embedded SHOUTcast and Icecast metadata, such as song title and artist, to Adobe Flash Player. The SHOUTcast example that's included with the Wowza Streaming Engine installation illustrates these capabilities.

Wowza Streaming Engine can deliver two-way video, audio, and text chat to Adobe Flash Player.

Live stream transcoding and transrating

The Transcoder in Wowza Streaming Engine is a real-time video transcoding and transrating solution that ingests a live stream, decodes the video and audio, and then re-encodes the stream for delivery to desired playback devices. It can decode and re-encode audio and video in multiple formats with keyframes that are properly aligned for adaptive bitrate delivery. The following are some common scenarios:

- **Transcode** – Ingest a non-H.264/VP8/VP9 video and non-AAC/MP3/Vorbis/Opus audio stream and convert it to a set of H.263, H.264, VP8, or VP9 video and AAC, Vorbis, or Opus audio renditions that have aligned key frames for adaptive bitrate streaming.

- **Transrate** – Ingest an H.264 video and AAC/MP3 audio stream and create a full set of bitrate renditions that have key frames aligned to the source stream for adaptive bitrate streaming.
- **Audio-only** – Ingest an H.264 video and Speex audio stream from Adobe Flash Player and convert the Speex audio format to AAC, Vorbis, or Opus to make the stream compatible with additional player technologies.

Video and audio codecs supported by Transcoder

Transcoder supports the following video and audio codecs:

Video (decoding)	Video (encoding)
H.264	H.263v2
MPEG-2	H.264
MPEG-4 Part 2	VP8
VP8	VP9
VP9	H.265/HEVC
Audio (decoding)	Audio (encoding)
AAC	AAC
G.711 (μ-law and A-law)	Vorbis
MPEG-1 Layer 1/2	Opus
MPEG-1 Layer 3 (MP3)	
Speex	
Vorbis	
Opus	

Transcoder and hardware acceleration

Transcoder, which runs on 64-bit Windows and Linux operating systems, can be configured to take advantage of hardware acceleration. Hardware acceleration is recommended but not required. If your configuration doesn't include hardware acceleration, a built-in software encoder is used.

Transcoder can use the following hardware acceleration technologies:

- **Intel Quick Sync Video** (for both accelerated video decoding and encoding) – For recommended workstation and server-level hardware specifications, and links to configuration instructions, see [Server specifications for Intel Quick Sync acceleration with Wowza Streaming Engine transcoding](#).
- **NVIDIA NVENC** (for accelerated video encoding *only*) and **NVIDIA CUDA/NVCUVID** (for accelerated video decoding *only*) – For a list of supported NVIDIA graphics cards that are compatible with Transcoder and links to configuration instructions, see [Server specifications for NVIDIA NVENC and NVIDIA CUDA acceleration with Wowza Streaming Engine transcoding](#).

Overlays on transcoded streams

With Transcoder, you can apply static GIF, JPEG, PNG, and BMP overlay images to streams and customize the location, size, alignment, and opacity of the image to achieve static image effects such as a watermark to your video. With Transcoder and the Wowza Streaming Engine Java API, you can also apply dynamic overlay images to streams. The Java API can be configured manually or pre-programmed based on external events, making it a powerful tool for adding premium TV-like experiences. See [Add graphic overlays to transcoded live streams in Wowza Streaming Engine](#).

For more information about Transcoder, see [About Wowza Streaming Engine Transcoder](#).

Live stream nDVR

The nDVR feature in Wowza Streaming Engine provides the ability to record a live stream into a cache on the server. Viewers that join the live stream in-progress can access the cache to rewind to the beginning of the live stream (or rewind within the part of the stream that you specify) and then use DVR playback controls in their player to watch the stream from that point forward. Configuration for client playback of recorded streams is similar to playback of live streams from Wowza Streaming Engine.

For more information about nDVR, see [About live stream nDVR](#).

Stream encryption with DRM

Wowza Streaming Engine integrates with third-party digital rights management (DRM) key management service partners to enable on-the-fly encryption of premium live and VOD content for a variety of playback devices. For live workflows, per-stream encryption is available with the ability to rotate keys. For VOD workflows, per-asset and per-session encryption is available with the ability to rotate keys. Both live and VOD key rotation support is available for HLS streaming.

Key management system integration is supported for the following providers:

- **BuyDRM KeyOS** – Provides Microsoft PlayReady encryption services for HLS, MPEG-DASH, and Microsoft Smooth Streaming and playback with BuyDRM players and Smooth Streaming clients on Windows, macOS, iOS and Android devices, Windows phones, game consoles, set-top boxes, and smart TVs.
- **EZDRM** – Provides Microsoft PlayReady encryption services for Smooth Streaming playback with Smooth Streaming clients on Windows, macOS, Windows phones, game consoles, set-top boxes, and smart TVs and with Discretix SecurePlayer media players on iOS and Android devices.
- **Verimatrix** – Provides Verimatrix VCAS and Microsoft PlayReady encryption services for HLS and Smooth Streaming playback with Verimatrix ViewRight and Smooth Streaming clients on Windows, macOS, iOS and Android devices, Windows phones, game consoles, set-top boxes, and smart TVs.

For more information about DRM, see [About digital rights management and Wowza Streaming Engine](#).

Note

Wowza Streaming Engine APIs can encrypt live and VOD HLS streams on the fly using SAMPLE-AES (sample-level encryption for version 5 of the HLS streaming protocol), ENVELOPE-PLAYREADY (supported by BuyDRM player technology with PlayReady DRM), and CHUNK-PLAYREADY (supported by INSIDE Secure player technology with PlayReady DRM). You can also use the Wowza Streaming Engine API to encrypt live and VOD Microsoft Smooth Streaming content on the fly with PlayReady protection for INSIDE Secure player technology. For more information, see [Secure HLS streaming using DRM encryption with Wowza Streaming Engine](#).

AddOns

A variety AddOn packages can be downloaded and installed to extend and enhance Wowza Streaming Engine functionality.

- **Dynamic Load Balancing** – Enables you to dynamically distribute HTTP, RTMP, and RTSP streams across multiple Wowza Streaming Engine edge servers. The edge servers communicate with one or more Wowza Streaming Engine load balancers, and clients connect to the load-balancing server to get the least-loaded edge server. See [Get the Dynamic Load Balancing AddOn for Wowza Streaming Engine](#).
- **Wowza StreamLock**– A security option for network encryption that provides near-instant provisioning of free 256-bit Secure Sockets Layer (SSL) certificates to verified Wowza users for use with Wowza Streaming Engine. StreamLock-provisioned SSL certificates provide the best security when used with RTMP. The certificates can also be used for secure HTTP streaming (HTTPS). Certificates expire after one year. See [Get SSL certificates from the Wowza Streaming Engine StreamLock service](#).

- **StreamNameAlias** – Enables you to simplify complex URL-based stream names with aliases, provide security to limit the valid stream names used, or map one stream name to another. See [Get the StreamNameAlias AddOn for Wowza Streaming Engine](#).
- **Wowza Load Test Tool** – Generate load on a single Wowza Streaming Engine instance to test configuration and performance of RTMP and HLS streaming. See [Get the Wowza Load Test Tool for Wowza Streaming Engine](#).

Installed examples

Wowza Streaming Engine includes the several examples that highlight various aspects of server functionality, including how to configure and play a live stream, how to configure and play VOD content, and how to set up live stream nDVR. You can find the examples in the **[install-dir]/examples** folder. The **README.html** file in that folder describes the examples and how to install them.

Server installation

Wowza Streaming Engine is a powerful Java-based media server. When you install Wowza Streaming Engine, it automatically installs the server version of Java that it requires. In addition, an installation wizard guides you through the process, making it easy to get Wowza Streaming Engine up and running.

Before you start

Before you install Wowza Streaming Engine be aware of these prerequisites, cautions, and best practices.

- Wowza Streaming Engine is built on Java 9 (OpenJDK Java SE JRE 9.0.4) and supports Java versions 9 - 12. Earlier versions of Java are not supported. For optimal performance and stability, we recommend using the version of Java that installs by default. To use Wowza Streaming Engine Manager, you *must* use the version of Java that installs with the server software. If you can't or don't want to use the version of Java that installs with Wowza Streaming Engine, see [Manually install and troubleshoot Java on Wowza Streaming Engine](#).
- If you have an earlier version of Wowza Streaming Engine installed and you want the current version installed, run the updater, not the installer. For instructions, see [Update your Wowza Streaming Engine installation](#).
- As a best practice, install and run only one version and instance of Wowza Streaming Engine on a single computer.
- To confirm that your computer meets the minimum system requirements for running Wowza Streaming Engine, see [Wowza Streaming Engine Specifications](#).
- If you're upgrading from Wowza Media Server™ software to Wowza Streaming Engine, you must uninstall Wowza Media Server before installing Wowza Streaming Engine. For instructions, see [Upgrade from Wowza Media Server to Wowza Streaming Engine](#).

- A silent installation option is available for all platforms, including Red Hat Package Manager and Debian Package Manager options for Linux. Approval is required for silent installations. For information, contact sales@wowza.com.

Installing Wowza Streaming Engine

Wowza Streaming Engine is installed using an install wizard on Windows, macOS, and Linux. The installer creates a new, clean instance of the Wowza Streaming Engine on the computer where it's installed.

1. If you don't already have the Wowza Streaming Engine installer on your system, go to the Wowza [Downloads webpage](#) and click **Download** for the installer for your desired operating system.
2. When the download completes, do one of the following:

- **Windows** – Double-click the file **WowzaStreamingEngine-4.7.8-windows-installer.exe**.
- **macOS** – Double-click the file **WowzaStreamingEngine-4.7.8-osx-installer.dmg**, and then double-click **WowzaStreamingEngine-4.7.8-osx-installer** package icon.
- **Linux** – Navigate to the folder that contains the downloaded package and then execute the following commands:

```
sudo chmod +x WowzaStreamingEngine-4.7.8-linux-x64-  
installer.run  
sudo ./WowzaStreamingEngine-4.7.8-linux-x64-installer.run
```

3. In the installation wizard, accept the terms of the license agreement.
4. Enter a valid license key.

If you acquired a new license key, you'll find it in the email that you received from Wowza Sales. If you have a previous version of Wowza Streaming Engine installed, look for the license key in the **[install-dir]/conf/Server.license** file.

5. Create a user name and password for an Administrator account. You'll use these credentials to sign in to the browser-based Wowza Streaming Engine Manager. The user name and password are case-sensitive.
6. Confirm or change the install location.

By default, Wowza Streaming Engine installs in the following locations:

- **Windows** – /Program Files (x86)/Wowza Media Systems/Wowza Streaming Engine 4.7.8/
- **macOS** –
 - /Applications/Wowza Streaming Engine 4.7.8/
 - /Library/LaunchDaemons/
 - /Library/WowzaStreamingEngine/ (an alias)
 - /Library/WowzaStreamingEngine-4.7.8/
- **Linux** – /usr/local/WowzaStreamingEngine-4.7.8/ (as the **root** user)

7. Accept the default option **Start Wowza Streaming Engine automatically**.

This option instructs the server software and Wowza Streaming Engine Manager to start automatically as system services. If you don't choose this option, you must start Wowza Streaming Engine and Wowza Streaming Engine Manager manually before you can use the software. See [Starting and stopping the software](#) and [Starting and stopping Wowza Streaming Engine Manager](#).

8. Click **Finish**.

Starting and stopping the software

Wowza Streaming Engine and Wowza Streaming Engine Manager can run as system services or in standalone mode.

System services start automatically when you start the computer and remain on until you turn them off. By default, Wowza Streaming Engine and Wowza Streaming Engine Manager install as system services, which means you're running an active instance of Wowza Streaming Engine from the moment of install and any time the host computer is on.

Standalone mode operates independently of the operating system; you start and stop the software on demand. As with any standalone software, if you forget or fail to quit the program, you're prompted to do so when you turn off the computer. Standalone mode is required for running Transcoder with accelerated hardware in Windows. It's also useful in testing environments because you can see log output in the console immediately.

You can, however, manually start and stop Wowza Streaming Engine at any time, in either operational mode. For example, Subscription license holders might want to turn off the software as a service to avoid being charged for inactive instances of Wowza Streaming Engine.

Wowza Streaming Engine can't run as a service and in standalone mode at the same time.

Start and stop Wowza Streaming Engine as a service (Windows)

To start the Wowza Streaming Engine service:

1. Press Win key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine 4.7.8** and then click **Start**.

To stop the Wowza Streaming Engine service:

1. Press Win key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine 4.7.8** and then click **Stop**.

Wowza Streaming Engine can be set to start automatically as a Windows service when Windows starts. To prevent the service from starting automatically when Windows starts:

1. Press WIN key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine 4.7.8**, and then click **Properties**.
3. In the **Properties** dialog box, on the **General** tab, set **Startup type** to **Manual**.

Start and stop Wowza Streaming Engine in standalone mode (Windows)

To start Wowza Streaming Engine in standalone mode, make sure that the Wowza Streaming Engine service is stopped (see above), and then do the following:

1. Press WIN key + R, type **cmd** in the **Run** dialog box, and then press Enter.
2. In the Command Prompt window, execute the following commands:

```
cd %WMSAPP_HOME%\bin
startup.bat
```

To stop the software:

1. Press WIN key + R, type **cmd** in the **Run** dialog box, and then press **Enter**.
2. In the Command Prompt window, execute the following commands:

```
cd %WMSAPP_HOME%\bin
shutdown.bat
```

Notes

- By default, the Wowza Streaming Engine 4.7.8 service runs under the Local System account. This can limit how Wowza Streaming Engine interacts with the underlying operating system. For example, you may have issues streaming content from UNC paths. To address this issue, update the service to run as a named user. To do this, right-click the service name in the **Services** window, click **Properties**, and then on the **Log On** tab specify an alternate user account that the service can use to log on under **This account**.

- The hardware acceleration used by Transcoder is only available when running Wowza Streaming Engine as a Windows standalone application. It's not available when Wowza Streaming Engine is invoked as a service.

Start and stop Wowza Streaming Engine as a service (macOS)

To start the service, double-click the **Start Services** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following command:

```
sudo launchctl load -w  
/Library/LaunchDaemons/com.wowza.WowzaStreamingEngine.plist
```

To stop the service, double-click the **Stop Services** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following command:

```
sudo launchctl unload -w  
/Library/LaunchDaemons/com.wowza.WowzaStreamingEngine.plist
```

Note

The **Start Services** and **Stop Services** applications also start and stop the Wowza Streaming Engine Manager system service. See [Starting and stopping Wowza Streaming Engine Manager](#).

Start and stop Wowza Streaming Engine in standalone mode (macOS)

To start the software, double-click the **Start Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.7.8/bin  
./startup.sh
```

To stop the software, double-click the **Stop Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.7.8/bin  
./shutdown.sh
```

Note

The **Start Standalone Mode** and **Stop Standalone Mode** applications also start and stop Wowza Streaming Engine Manager in standalone mode. See [Starting and stopping Wowza Streaming Engine Manager](#).

Start and stop Wowza Streaming Engine as a service (Linux)

Note

The operations in this section must be performed as the **root** user with **sudo** access.

To start the service, open a Terminal window and enter one of the following commands, depending on your Linux distribution:

```
sudo service WowzaStreamingEngine start
```

-or-

```
/etc/init.d/WowzaStreamingEngine start
```

To stop the service, enter:

```
sudo service WowzaStreamingEngine stop
```

-or-

```
/etc/init.d/WowzaStreamingEngine stop
```

Notes

- If these instructions don't apply to your Linux distribution, consult your Linux manual.
- The Linux services script subsystem doesn't use the full \$PATH definition to determine the location of Linux commands. It uses what's known as the **init** path. This can lead to an issue on Linux distributions where the default installation location for Java can't be found by applying the **init** path. See [Manually install and troubleshoot Java on Wowza Streaming Engine](#).

Start and stop Wowza Streaming Engine in standalone mode (Linux)

To start the software, open a Terminal window and enter the following commands:

```
cd /usr/local/WowzaStreamingEngine/bin  
./startup.sh
```

To stop the software, enter:

```
cd /usr/local/WowzaStreamingEngine/bin  
./shutdown.sh
```

Uninstalling Wowza Streaming Engine

- **Windows** – Go to the **Programs and Features Control Panel**, click **Wowza Streaming Engine 4.7.8**, and then click **Uninstall**.
- **macOS** – Go to **/Applications/Wowza Streaming Engine 4.7.8** and double-click **Wowza Streaming Engine Uninstall**.
- **Linux** – Run the following commands:

```
cd usr/local/WowzaStreamingEngine
sudo ./uninstall
```

Running Wowza Streaming Engine as a named user

On macOS and Linux, the default installation of Wowza Streaming Engine runs the server as the **root** user. If you want to run the server as a different user, follow the instructions in [Run Wowza Streaming Engine as a named user \(Linux and macOS\)](#) to create a new user and configure the server to run as that user.

Note

For security reasons, the non-**root** user can't bind to port numbers less-than or equal to 1024 on most Linux and Unix distributions. If you plan to run Wowza Streaming Engine on a lower-numbered port such as 80 (HTTP), 443 (HTTPS, RTMPS), or 554 (RTSP), Wowza Streaming Engine must run as the **root** user.

Entering a license key

The license key you enter when you install Wowza Streaming Engine is displayed in the **License Keys** box in Wowza Streaming Engine Manager. If you switch your licensing option for the Wowza Streaming Engine instance, you can replace the existing license key with the new license key without reinstalling the software. For example, if you purchased license keys to enable Transcoder, nDVR, and DRM for use with the licensed server instance, you can add these license keys. All license key values are stored in the **Server.license** file located in the Wowza Streaming Engine installation directory **[install-dir]/conf/**:

- **Windows** – %WMSCONFIG_HOME%\conf\Server.license
- **macOS** – /Library/WowzaStreamingEngine/conf/Server.license
- **Linux** – /usr/local/WowzaStreamingEngine/conf/Server.license

To add license keys in Wowza Streaming Engine Manager, do the following:

1. Click the **Server** tab, and then click **Server Setup** in the contents panel.
2. On the **Server Setup** page, click **Edit**.

3. In the **License Keys** box, enter your license key for Wowza Streaming Engine. If you need to enter multiple license keys, enter each license key on a separate line.
4. Click **Save**, and then click **Restart Now** at the top of the **Server Setup** page when prompted. The new license(s) take effect after the server restarts.

Notes

- After you restart the server, Wowza Streaming Engine Manager displays the first and last five digits of the license keys that you entered in the **License Keys** box to help protect this information.
- You can also open the **Server.license** file in a text editor, enter each new license key on a new line, and then restart the server.

Perpetual and Subscription licenses

Perpetual and Subscription licenses for Wowza Streaming Engine provide for unlimited connections to the server software instance and enable use of Transcoder, nDVR, and DRM technologies that are integrated with each licensed instance.

A Perpetual license is best for stable, long-term demand. A Perpetual license key purchased after December 22, 2015 has an **EPBP4** prefix and is for use with one Wowza Streaming Engine instance and the integrated Transcoder, nDVR, and DRM technologies.


Server Setup

The screenshot shows the 'Server Setup' page in the Wowza Streaming Engine Manager. At the top, there are three tabs: 'Basic', 'Properties', and 'Server Listeners'. Below the tabs is an 'Edit' button. The 'Name' field is labeled 'Wowza Streaming Engine'. The 'Description' field contains the text: 'Wowza Streaming Engine is robust, customizable, and scalable server software that powers reliable streaming of high-quality video and audio to any device, anywhere.' The 'License Keys' field is highlighted with a red box and contains the text: 'EPBP4-XXXXX-XXXXX-XXXXX-h5t3C'. Below this is the 'Default Stream Prefix' field, which is labeled 'mp4'. The 'Options' section has two checkboxes: 'Enable monitoring and statistics for this server' (checked) and 'Allow RTP datagram port sharing' (unchecked). At the bottom, the 'RTP Datagram Starting Port' is labeled '6970'.

For details, see [Wowza Streaming Engine Pricing](#).

A Subscription license is best for variable demand. You can install as many instances of Wowza Streaming Engine as needed using the same license key and enable the Transcoder, nDVR, and DRM technologies integrated with each instance. A Subscription license key has an **ENGM4** prefix.

Server Setup

 Edit

Name
Wowza Streaming Engine

Description
Wowza Streaming Engine is robust, customizable, and scalable server software that powers reliable streaming of high-quality video and audio to any device, anywhere.

License Keys
ENGM4-XXXXX-XXXXX-XXXXX-XXXXX-yy6vu

Default Stream Prefix
mp4

Options
☒ Enable monitoring and statistics for this server
☐ Allow RTP datagram port sharing

RTP Datagram Starting Port
6970

For details, see [Wowza Streaming Engine Pricing](#).

Notes

- If you purchased a Perpetual license for Wowza Streaming Engine before December 23, 2015, and the license key has an **EPBU4** prefix, it licenses Wowza Streaming Engine and the Transcoder and nDVR technologies. A separate license key is provided to enable the Wowza DRM technology integrated with the server instance (you don't have to enter the DRM license key in the **License Keys** box unless you want to enable this technology in Wowza Streaming Engine).
- If you purchased a license for Wowza Streaming Engine before January 1, 2015, contact sales@wowza.com to learn how to license the software.

Trial licenses

A Trial license for Wowza Streaming Engine lets you try out the software for free. After it expires and you acquire a new, paid license for the software, you must delete the Trial license key from the **License Keys** box and then add the new, paid key. If you're not sure if a license key is a Trial key, you can find it in the email that you received from Wowza when you downloaded the trial software.

Note

You don't need to reinstall Wowza Streaming Engine or re-create its settings when you replace the Trial license key with a paid license for the software.

Ports used for streaming





Before streaming with Wowza Streaming Engine, open the required ports on your firewall. The following table shows the default ports that Wowza Streaming Engine uses for streaming. All of these port numbers are configurable.

- **TCP 1935** – RTMP/RTMPE/RTMPT/RTSP-interleaved streaming/WOWZ™
- **TCP 8086-8088** – Administration
- **UDP 6970-9999** – RTP UDP streaming

By default, Wowza Streaming Engine is configured to use only TCP port 1935 for streaming. You may want to configure additional ports for streaming such as TCP port 80 for HTTP or RTMPT streaming or TCP port 554 for RTSP streaming. To add an additional port, go to the **Virtual Host Setup** page in Wowza Streaming Engine Manager and edit the **Default Streaming** host port.

Host Ports

[Add Host Port...](#)

Name	Type	IP Address	Port	SSL/StreamLock Enabled	Actions
Default Streaming	Streaming	*	1935	false	 
Default Admin	Admin	*	8086	false	 

In the **Edit host port** dialog box, add the additional ports to the **Port(s)** list (this list is comma-delimited).

Edit host port X

Name
Default Streaming

Type
Streaming

IP Address *

Port(s) *

1935,80

Comma-separated list

☐ Enable SSL/StreamLock

Keystore Path (StreamLock certificate path)

Keystore Password (StreamLock certificate password)

Cancel Apply

Wowza Streaming Engine can't share ports with other programs or services, so make sure no other programs or services are running on the added ports.

The following table shows some of the common ports used for streaming.

- **TCP 80** – HLS, MPEG-DASH, HDS, Smooth Streaming, RTMPT
- **TCP 443** – RTMPS, HTTPS
- **TCP 554** – RTSP

Server configuration and tuning

Wowza Streaming Engine configuration settings are stored in a set of XML configuration and properties files in the **[install-dir]/conf** folder. The settings can be changed by configuring options and properties in Wowza Streaming Engine Manager or by editing them in a text editor. If you choose to manage Wowza Streaming Engine configuration settings by editing the XML files directly, be sure to review the [Wowza Streaming Engine Configuration Reference Guide](#), which describes the most commonly used configuration settings.

The following configuration files are read when the server starts:

Server configuration files

- **MediaCache.xml** – Media Cache configuration
- **Server.xml** – General server configuration
- **Tune.xml** – Server performance tuning configuration
- **VHosts.xml** – Virtual hosts definition
- **Log4j.properties** – Logging configuration

VHost configuration files

- **StartupStreams.xml** – Streams started at virtual host startup
- **VHost.xml** – Virtual host configuration
- **VHosts.xml** – Virtual hosts configuration

Application configuration files

- **Application.xml** – Application configuration

Wowza Streaming Engine is automatically tuned to take best advantage of available hardware resources when the server starts. The server calculates an appropriate Java heap size, garbage collection (GC) settings, and other Java command-line options based on available hardware, the computer and Java Virtual Machine (JVM) bitness, and other factors.

By default, Wowza Streaming Engine sets the Java heap size to a value that's suitable for application development environments. Before you deploy the server in production environments where it may use memory extensively when heavily loaded, you can select an option in Wowza Streaming Engine Manager that automatically sets the heap size to a predefined value that's appropriate for production use. You can also adjust many other performance settings from the default values that are calculated by the server in Wowza Streaming Engine Manager to fine-tune the server's performance. See [Tune Wowza Streaming Engine for optimal performance](#).

Software updates

Between production releases, development builds are periodically released in the form of updates. Updates give you early access to new features and the opportunity to give feedback. Information about what's included in each update is in a **README.txt** file in the updater .zip file. For information about how to apply an update, see [Update your Wowza Streaming Engine installation](#).

Application configuration

Wowza Streaming Engine controls streaming through *applications*. A application can be configured to deliver either live or VOD content over multiple protocols for playback on a variety of screens and devices.

It's easy to define an application in Wowza Streaming Engine Manager. For example, to create a new application named **myapplication**, do the following:

1. Click the **Applications** tab in Wowza Streaming Engine Manager and then click **Add Application** in the contents panel.
2. On the **Add Application** page, review the content in the **Help** panel to decide what type of application you want to create.
3. Click the application type in the **Add Application** page.
4. In the **New Application** dialog box, enter the name **myapplication** and then click **Add**.

The **myapplication** page is displayed, and you can configure the application settings.

A single application can be configured to deliver a live or VOD stream at the same time to desktop browsers; iOS and Android mobile devices; Apple TV and Roku and Amino set-top boxes; and MPEG-DASH, Flash, Microsoft Silverlight, and RTSP/RTP-based players. See the [Connect live sources](#) page of the Wowza Streaming Engine documentation website for links to numerous articles on how to configure applications for common streaming scenarios.

Let's explore the nitty gritty details of application configurations.

Applications and application instances

An **Application.xml** file defines the configuration that you set up in Wowza Streaming Engine Manager for a given application. An application instance is an instantiation of an application and provides a namespace and context for streaming. An application instance is started dynamically; a single application can have multiple named application instances running simultaneously. If no name is specified for an application instance, then the default name **_definst_** is used. In many streaming scenarios, a single application instance is used per-application and the name is never referenced and defaults to **_definst_**. It's more common to use multiple application instances in video chat and video conferencing scenarios where you must create multiple rooms for streaming. In this case, application instances are used to separate streaming into rooms. Each room is a separate application instance, which provides separation and a namespace for each room.

When an application instance loads, it looks for an **Application.xml** file in **[install-dir]/conf/** and **[install-dir]/conf/[application name]/**. The first **Application.xml** file that's found is used.

Playback URL formats

All streaming in Wowza Streaming Engine is initiated with a Uniform Resource Locator (URL). The application and application instance names are specified as part of the streaming URL. The URL formats used for streaming, whether for desktop browsers, iOS and Android devices, or MPEG-DASH or other players, follow a similar format:

```
[protocol]://[address]:[port]/[application]/[appInstance]/[streamName]/[post-fix]
```

Where:

- **[protocol]** is the streaming protocol (http, rtmp, rtsp, and so on)
- **[address]** is the address of the server running Wowza Streaming Engine
- **[port]** is the port number to use for streaming (1935 is the default)
- **[application]** is the application name
- **[appInstance]** is the application instance name
- **[streamName]** is the stream name and prefix
- **[post-fix]** is optional information specific to player technology

In most streaming scenarios, if **[streamName]** doesn't have path elements and the default **[appName]** name is used, the URL can be shortened to:

`[protocol]://[address]:[port]/[application]/[streamName]`

The following are example URLs for various streaming protocols. The examples assume that a live video with the stream name **myStream** using the application name **live** is streamed.

HLS

`http://mycompany.com:1935/live/myStream/playlist.m3u8`

MPEG-DASH

`http://mycompany.com:1935/live/myStream/manifest.mpd`

HDS

`http://mycompany.com:1935/live/myStream/manifest.f4m`

Smooth Streaming

`http://mycompany.com:1935/live/myStream/Manifest`

RTMP

Server: `rtmp://mycompany.com/live`

Stream: `myStream`

RTSP/RTP

`rtsp://mycompany.com:1935/live/myStream`

Stream types

Named stream types control different types of streaming (live, VOD, recording, origin/edge, and so on). Stream types are automatically configured when you create an application and configure it in Wowza Streaming Engine Manager. Alternatively, you can use a text editor to change the **StreamType** property in the **<Streams>** container element in an **Application.xml**. Here are all of the stream types and their uses.

Stream type	Description
default	VOD
file	VOD
live	Deliver live streams; best for one-to-many streaming of live events
live-lowlatency	Deliver live streams over RTMP; best for one-to-one or one-to-few video/audio chat applications
live-record	Same as live but the stream is also recorded
live-record-lowlatency	Same as live-lowlatency but the stream is also recorded
liverepeater-edge	Deliver live streams across multiple Wowza Streaming Engine servers in an origin/edge configuration; use to configure the edge application(s)
liverepeater-edge-lowlatency	Deliver live streams across multiple servers in an origin/edge configuration; use to configure edge application(s) when low latency is important
liverepeater-edge-origin	Deliver live streams across multiple Wowza Streaming Engine servers in an origin/edge/edge configuration; use to configure a middle-edge application
record	Video recording
rtp-live	Re-stream RTSP/RTP, native RTP, or MPEG-TS streams
rtp-live-lowlatency	Re-stream RTSP/RTP, native RTP, or MPEG-TS streams when low latency is important
rtp-live-record	Same as rtp-live but the stream is also recorded
rtp-live-record-lowlatency	Same as rtp-live-lowlatency but the stream is also recorded
shoutcast	Re-stream SHOUTcast/Icecast MP3 or AAC+ audio streams
shoutcast-record	Same as shoutcast but the stream is also recorded

Each stream type exposes properties that are used for tuning the stream type. For example, the stream type definitions for **live** and **live-lowlatency** differ only in the tuning that's accomplished through the stream properties. Defined properties for a stream type can be overridden on a per-application basis by defining new property values on an application's **Properties** tab in Wowza Streaming Engine Manager or by editing the **<Streams>/<Properties>** container element in **Application.xml**.

HTTP streamers and live stream packetizers

HTTP streamers define the streams in an application (live or VOD) that are available for playback to different player technologies. In Wowza Streaming Engine Manager, you can select one or more of the following **Playback Types** options for an application. When selecting multiple options, the corresponding HTTP streamers are added to the **<HTTPStreamers>** section in **Application.xml** as a comma-separated list.

Playback type	Description
HLS	Enables the application to stream live and VOD content to iOS devices, QuickTime player, Safari, and to other devices that support the HLS protocol. Adds the cupertinostreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
MPEG-DASH	Enables the application to stream live and VOD content to players that support the DASH protocol. Adds the mpegdashstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
HDS	Enables the application to stream live and VOD content to Flash Player using the HDS protocol. Adds the sanjosestreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
Smooth Streaming	Enables the application to stream live and VOD content to Microsoft Silverlight using the Smooth Streaming protocol. Adds the smoothstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
nDVR (live streaming only)	When you enable nDVR for live or live http origin applications in Wowza Streaming Engine Manager, enables the application to stream live content from Wowza Streaming Engine (origin) to Wowza Streaming Engine (edge). Adds the dvrchunkstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .

Note

For CMAF streaming, which can't be configured in Wowza Streaming Engine Manager, both the **cupertinostreaming** and **mpegdashstreaming** HTTP streamers must be specified in **Application.xml**. For information on how to create a basic CMAF stream, see [Stream using CMAF with Wowza Streaming Engine](#).

Live streams coming into Wowza Streaming Engine must be packaged (*packetized*) for delivery to players that use HTTP streaming protocols. The **<Streams>/<LiveStreamPacketizers>** section in **Application.xml** specifies the streaming protocols that are used when packetizing live streams. There are two types of packetizers: streaming packetizers and repeater packetizers.

Streaming packetizers are used when delivering a live stream from a single Wowza Streaming Engine server to clients. They're also used to deliver a live stream from a Wowza Streaming Engine origin server to an edge server when using the live repeater mechanism in an origin/edge configuration. When you select **Playback Types** options in Wowza Streaming Engine Manager to create HTTP streamers for live applications, the corresponding live stream packetizer values (separated by commas) are added to the **<LiveStreamPacketizers>** section in **Application.xml**.

Playback type	Description
HLS	Enables HLS live streaming. Adds the cupertinostreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
MPEG-DASH	Enables MPEG-DASH live streaming. Adds the mpegdashstreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
HDS	Enables HDS live streaming. Adds the sanjosestreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
Smooth Streaming	Enables Smooth Streaming live streaming. Adds the smoothstreaming streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
nDVR	Adds the dvrstreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml for nDVR support.

Note

For CMAF streaming, which can't be configured in Wowza Streaming Engine Manager, the **cmafstreamingpacketizer** live stream packetizer must be specified in **Application.xml**. For information on how to create a basic CMAF stream, see [Stream using CMAF with Wowza Streaming Engine](#)

Repeater packetizers deliver a live stream from a Wowza Streaming Engine edge server to clients in a live stream repeater (origin/edge) configuration. When you select **Playback Types** options in Wowza Streaming Engine Manager to create HTTP streamers for live edge applications, the corresponding repeater packetizer values (separated by commas) are added to the **<LiveStreamPacketizers>** section in **Application.xml**.

Playback type	Description
HLS	Enables HLS live stream repeating. Adds the cupertinostreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
MPEG-DASH	Enables MPEG-DASH live stream repeating. Adds the mpegdashstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
HDS	Enables HDS live stream repeating. Adds the sanjosestreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
Smooth Streaming	Enables Smooth Streaming live stream repeating. Adds the smoothstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
nDVR	Adds the dvrstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml for use with nDVR.

For information about how to implement live stream repeating (origin/edge configurations) in Wowza Streaming Engine, see [Live stream repeater \(origin/edge live streaming\)](#). For information about how to configure CMAF stream repeating, see [Stream using a CMAF live stream repeater in Wowza Streaming Engine](#).

Note

nDVR in Wowza Streaming Engine provides the ability to record a live stream while simultaneously allowing users to play or pause the live stream, rewind to a previously recorded point, or resume viewing at the live point. nDVR can be extended to an edge server in an origin/edge configuration. See [Set up a Wowza Streaming Engine live stream repeater for Wowza nDVR](#).

Timed text providers

Wowza Streaming Engine supports timed text (closed captioning) for live and VOD streams. It converts caption data from a variety of instream and file-based sources to appropriate caption formats for live and on-demand video streaming using the HDS, HLS, and RTMP streaming protocols. This feature helps US broadcasters to comply with the Twenty-First Century Communications and Video Accessibility Act of 2010 and increasing requirements in the European Union by providing captioning for television programs that are distributed over the Internet.

Closed captioning for live streams

For live streams, Wowza Streaming Engine can ingest instream closed caption information from CEA-608 data or AMF onTextData events. These ingested captions can be delivered as CEA-608-formatted SEI data in HLS streams or as onTextData events in HDS and RTMP streams. In addition, in-stream CEA-608 caption data can be passed through Transcoder for delivery in HLS streams. For live and live edge applications in Wowza Streaming Engine Manager, you can configure the following **Closed Caption Sources** options to enable the application to ingest the caption data.

Live closed caption source	Description
onTextData events in live streams	Enables the application to monitor live streams for Action Message Format (AMF) onTextData captions, decode the captions, and convert them to CEA-608-formatted SEI data in HLS streams. Adds the ModuleOnTextDataToCEA608 module to the <Modules> section in Application.xml .
Embedded CEA-608 captions in live streams	Enables the application to monitor live streams for CEA-608 captions, decode the captions, and convert them to onTextData events in HDS and RTMP streams. Adds the ModuleCEA608ToOnTextData module to the <Modules> section in Application.xml .
onCaptionInfo events in live streams	Specified by the onCaptionInfo events in live streams option. Enables the application to monitor live streams for AMF onTextData events and pass them through in HDS and RTMP streams. Adds the captionLiveIngest property to the <TimedText>/<Properties> section in Application.xml .

For more information, see [Configure closed captioning for Wowza Streaming Engine live streams](#).

Closed captioning for video-on-demand streams

For VOD streams, Wowza Streaming Engine can extract caption data from 3GPP Timed Text data embedded in MP4 files or use caption files in a variety of formats including Timed Text Markup Language (.ttml), SubRip Text (.srt), Scenarist Closed Caption (.scc), and Web Video Text Tracks (.vtt). Ingested captions can be delivered as CEA-608-formatted SEI data in HLS streams or as Action Message Format (AMF) onTextData events in HDS and RTMP streams. For VOD and VOD edge applications in Wowza Streaming Engine Manager, you can select one or more of the following **Closed Caption Sources** options to ingest caption data. When selecting multiple options, the corresponding timed text providers are added to the **<TimedText>** section in **Application.xml** as a comma-separated list.

VOD closed caption source	Description
Embedded 3GPP/MPEG-4 Timed Text tracks	Enables the application to pull captions directly from 3GPP tracks (codeID "tx3g") that are embedded in MP4 VOD assets. Enabled by default. Adds the vodcaptionprovidermp4_3gpp timed text provider to the <TimedText> section in Application.xml .
Timed Text (TTML/DXFP) file	Enables the application to pull captions from an external TTML-formatted caption file that sits next to the VOD asset in the application's content directory. Adds the vodcaptionproviderttml timed text provider to the <TimedText> section in Application.xml .
SubRip (SRT) file	Enables the application to pull captions from an external SRT-formatted caption file that sits next to the VOD asset in the application's content directory. Adds the vodcaptionprovidersrt timed text provider to the <TimedText> section in Application.xml .
Web Video Text Track (WebVTT) file	Enables the application to pull captions from an external WebVTT-formatted caption file that sits next to VOD asset in the application's content directory. Adds the vodcaptionproviderwebvtt timed text provider to the <TimedText> section in Application.xml .
Scenarist Closed Caption (SCC) file	Enables the application to pull captions from an external SCC-formatted caption file that sits next to VOD asset in the application's content directory. Adds the vodcaptionproviderscc timed text provider to the <TimedText> section in Application.xml .

See [Configure closed captioning for Wowza Streaming Engine video-on-demand streams](#).

Transcoder and nDVR configurations

The <Transcoder> and <DVR> container elements in an **Application.xml** file serve to configure a Wowza Streaming Engine application to use Transcoder and nDVR. See the [Wowza Streaming Engine Configuration Reference Guide](#) and these articles:

- [Set up and run Transcoder in Wowza Streaming Engine](#)
- [Set up and run Wowza nDVR in Wowza Streaming Engine](#)

Modules

Modules are Java classes that are loaded dynamically when an application instance is loaded and provide an application's functionality. In Wowza Streaming Engine Manager, the **Modules** list defines an order-dependent list of modules to load for a given application. Many AddOn packages provide additional functionality through the use of modules. See [Use Wowza Streaming Engine Java modules](#).

In Wowza Streaming Engine Manager, click the **Modules** tab on an application page to see the list of modules that are loaded.

[Setup](#)
[Properties](#)
[Modules](#)

Note: Items on this page should be configured by advanced users only.

Modules Java classes that extend an application's functionality. The list below defines an order-dependent list of modules to be loaded for a given application. The modules are loaded dynamically when the application instance is loaded. The base (ModuleCore) module must be included by the application for it to operate properly.

[Edit](#)

Name	Description	Fully Qualified Class Name
base	Base	com.wowza.wms.module.ModuleCore
logging	Client Logging	com.wowza.wms.module.ModuleClientLogging
flvplayback	FLVPlayback	com.wowza.wms.module.ModuleFLVPlayback
ModuleCoreSecurity	Core Security Module for Applications	com.wowza.wms.security.ModuleCoreSecurity

Each module must have a unique **Name**. The **Description** provides a detailed description of the module and isn't used in any operations. The **Class** is the fully qualified path to the Java class that provides the module's functionality. In general, new modules are always added to the end of the **Modules** list. The Java class that makes up a server-side module is most often bound to a .jar file in the Wowza Streaming Engine installation. Wowza Streaming Engine comes with many modules that can be added to the **Modules** list to provide additional functionality. For information, see [Use Wowza Streaming Engine Java modules](#). You can also use the Wowza IDE to develop your own custom modules to provide additional functionality. See [Extend Wowza Streaming Engine using the Wowza IDE](#).

Notes

- Access to the **Modules** tab is limited to administrators with advanced permissions. See [Managing sign-in credentials](#).
- Wowza provides an assortment of ready-to-use utility modules for Wowza Streaming Engine applications. For a complete list, see [Module examples](#).

Properties

Properties are name/value pairs that provide a means for tuning and modifying the default application configuration. Properties can also be used server-side as a means to pass data to custom modules from applications. Properties are listed in the **Application.xml** configuration file and are defined using the format:

```
<Property>
  <Name>[name]</Name>
  <Value>[value]</Value>
  <Type>[type]</Type>
</Property>
```

Where **<Name>** is the property name, **<Value>** is the property value, and **<Type>** is the property type. Valid property types are **Boolean**, **Integer**, **Long**, and **String**.

In Wowza Streaming Engine Manager, click the **Properties** tab on an application page and enable default properties to either add them to the application configuration or to override existing property values. For details about the properties, see the [Wowza Streaming Engine Configuration Reference Guide](#).

Many technical articles on the Wowza website explain how to use custom properties to tune Wowza Streaming Engine and to add advanced functionality. When adding custom properties, it's important to add them to the correct **<Properties>** container element in **Application.xml**. Article instructions always specify the **Path** value to use in the **Add Custom Property** dialog box, which adds the property to the correct **<Properties>** container.

Add Custom Property
X

Path

/Root/Application
▼

Name *

Type

String
▼

Value *

Cancel

+ Add

Note

Access to the **Properties** tab is limited to administrators with advanced permissions. See [Managing sign-in credentials](#).

Media types

Media types aren't defined in application configuration files but are an important part of streaming. Wowza Streaming Engine supports many media types. It can read the following media or file types:

- **MP4** (QuickTime container - .mp4, .f4v, .mov, .m4a, .m4v, .mp4a, .mp4v, .3gp, .3g2, etc.)
- **FLV** (Flash Video - .flv)
- **MP3** content (.mp3)
- **SMIL** (Synchronized Multimedia Integration Language - .smil)
- **AMLST** (API-based MediaList)

Media types are specified by a prefix to the stream name. For example, to play the MP4 file **mycoolvideo.mov**, use the stream name **mp4:mycoolvideo.mov**, where mp4: is the media type prefix. If no media type prefix is specified, the media type prefix defaults to mp4:. The following table shows the supported media type prefixes.

Media type prefix	Description
mp4:	QuickTime container (default if no prefix specified)
flv:	Flash Video
mp3:	MP3 file
id3:	MP3 file (returns only ID3 tag information)
smil:	Synchronized Multimedia Integration Language (for adaptive bitrate delivery)
ngrp:	Named Group (for adaptive bitrate delivery)
amlst:	API-based MediaList (for adaptive bitrate delivery)

The media type prefix is also used to control the file container that stores recorded live video. When publishing video, if the mp4: media type prefix is specified or if no prefix is specified, then the content is recorded to an MP4 (QuickTime) container. Only H.264, AAC, and MP3 content can be recorded to an MP4 container. If the flv: media type prefix is specified, an FLV (Flash Video) container is used.

Synchronized Multimedia Integration Language (.smil) files provide a means to specify a group of live streams or VOD files for adaptive bitrate switching. For stream switching to occur correctly, key frames must be properly aligned across all of the available bitrates in a live stream. For VOD, multiple files must be pre-encoded to the desired bitrates and have key frames that are aligned across all of the encoded files. The smil: media type prefix is used to playback the content that's specified in .smil files.

Transcoder uses a templating system to cluster streams into logical groups (called *stream name groups*) for live adaptive bitrate delivery. Stream name groups and SMIL files serve the same purpose and either method can be used for playback of live streams. Stream name groups are defined in the transcoder template and are available for playback using the ngrp: media type prefix.

The Wowza Streaming Engine Java API can be used to intercept requests for adaptive bitrate streams and provide the stream grouping. To use this feature, you must use the amlst: stream name prefix to use a set of Java objects that describe the adaptive bitrate stream (an *API-based MediaList*). When Wowza Streaming Engine reads a SMIL file, it creates a MediaList and passes it back to the streaming provider. The API provides a means for intercepting the requests and creating the MediaList dynamically in a Wowza Streaming Engine module. See [Use Java API calls to resolve SMIL file requests \(AMLST\)](#).

Content storage

By default, Wowza Streaming Engine streams VOD content from (and records VOD content to) the **[install-dir]/content** folder. You can specify a different storage location for a VOD application in Wowza Streaming Engine Manager by changing the **Content Directory** value for the application. For example, to configure an application to use an application-specific content folder, select the **Application-specific directory** option:

Content Directory

☐ Use default

`${com.wowza.wms.context.VHostConfigHome}/content`

☒ Application-specific directory

`${com.wowza.wms.context.VHostConfigHome}/content/vod`

☐ Use the following directory:

Using this setting, content is streamed from the **[install-dir]/content/[application]** folder, where **[application]** is the application's name (**vod**).

Files that are required for streaming live content, such as Session Description Protocol (SDP) or .stream files, are also stored in the **[install-dir]/content** folder by default. You can specify a different storage location for a live application in Wowza Streaming Engine Manager by changing the **Streaming File Directory** value for the application. For example, to configure an application to use an application-specific folder, select the **Application-specific directory** option:

Streaming File Directory *

☐ Use default

`${com.wowza.wms.context.VHostConfigHome}/content`

☒ Application-specific directory

`${com.wowza.wms.context.VHostConfigHome}/content/live`

☐ Use the following directory:

Using this setting, the files can be accessed from the **[install-dir]/content/[application]** folder, where **[application]** is the application's name (**live**).

You can further customize content storage for your applications by specifying the fully qualified path to the storage location in the **Use the following directory** box. You can substitute variables in place of path elements. The following variables are supported:

- `${com.wowza.wms.AppHome}` – Application home directory
- `${com.wowza.wms.ConfigHome}` – Configuration home directory
- `${com.wowza.wms.context.VHost}` – Virtual host name
- `${com.wowza.wms.context.VHostConfigHome}` – Virtual host configuration directory
- `${com.wowza.wms.context.Application}` – Application name
- `${com.wowza.wms.context.ApplicationInstance}` – Application instance name

Advanced configuration

Wowza Streaming Engine offers a multitude of built-in advanced configuration options. Some advanced functionality is also provided by [AddOns](#).

MediaCaster, Stream files, and Startup Streams

MediaCaster is a system for re-streaming live streams. MediaCaster re-streams IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, streaming output from native RTP or MPEG-TS encoders, and RTMP streams from another Wowza Streaming Engine server (**liverepeater** streams). MediaCaster pulls a stream from a stream source and makes it available for streaming to all player technologies supported by Wowza Streaming Engine. The system works on demand: When the first request is received from a player for a given stream, a connection is made to the source stream and the stream is then made available to the player. When the last player stops watching the stream, MediaCaster waits for a timeout period. If no other players request the stream, the stream is stopped and isn't available for streaming until another request is made.

This on-demand startup functionality works great for RTMP and RTSP/RTP streaming where advanced packetization isn't required. However, it doesn't work for the HTTP streaming protocols (HLS, MPEG-DASH, HDS, and Smooth Streaming). HLS streams require about 30 seconds of video to be pre-packetized before playback can start, and Microsoft Silverlight clients require three times the key frame duration. Therefore, the stream must be started before it's ready for streaming over HTTP. Wowza Streaming Engine Manager provides features to start receiving MediaCaster streams and to keep them running.

Stream files

An easy method for re-streaming live MediaCaster streams is to configure a Stream file (a file with a .stream file name extension) that live applications can use to connect to the source stream through the MediaCaster system. A Stream file just contains the URI of the source stream. When the source stream is started, a live application can use the information in the Stream file to connect to the stream so that it's available for playback when requested by players.

As an example, to create a Stream file named **mycoolevent.stream**, do the following:

1. Click the **Server** tab in Wowza Streaming Engine Manager and then click **Stream Files** in the contents panel.
2. On the **Virtual Host Stream Files** page, click **Add Stream File**.
3. In the **Add Stream File** dialog box, enter the name **mycoolevent** and then click **Add**. The **mycoolevent.stream** page is displayed.
4. In **Stream URI**, enter the source stream URI and then click **Save**. For example, if you're using an MPEG-TS encoder, the URI value might be **udp://0.0.0.0:10000**.
5. Click **Return to Stream Files**.

Virtual Host Stream Files > mycoolevent.stream

defaultVHost

[← Return to Stream Files](#)

Basic Properties

Configure your Stream file below.

[Edit](#)

Stream URI
udp://0.0.0.0:10000

6. Click the **Connect** icon for **mycoolevent.stream**.

Virtual Host Stream Files

defaultVHost

Stream Files

[Copy Stream File](#) [+ Add Stream File](#)

Name	Actions
mycoolevent.stream	→ + ✎ 🗑

7. In the **Connect a Stream File** dialog box, configure the options to enable a live application to connect to the stream and then click **OK**.

Connect a Stream File

Stream Name
mycoolevent.stream

Application Name
live

Application Instance
☒ Connect to default application instance: _definst_
☐ Connect to application instance:

 Enter an existing application instance name. The application instance will be created if it does not exist.

MediaCaster Type
rtp

Cancel OK

Select the **MediaCaster Type** in the list that corresponds to the source stream type:

- **rtp** – For IP camera streams (RTSP/RTP streams) and for streams from native RTP and MPEG-TS encoders
- **shoutcast** – For SHOUTcast/Icecast streams
- **liverepeater** – For RTMP streams pulled from another Wowza Streaming Engine server

8. Click **OK** and restart Wowza Streaming Engine.

Note

In Wowza Streaming Engine Manager, you can connect to live MediaCaster streams that are referenced in Synchronized Multimedia Integration Language (SMIL) files. In the **Incoming Streams** feature, you can connect to MediaCaster streams to record them.

Startup Streams

The second method for starting live MediaCaster streams is to use **Startup Streams** in Wowza Streaming Engine Manager to create stream entries in the **[install-dir]/conf/StartupStreams.xml** file. Stream entries in this file start automatically when the

server starts (or more specifically, when a virtual host starts). The format of a single entry in **StartupStreams.xml** is:

```
<StartupStream>
  <Application>[application]</Application>
  <MediaCasterType>[mediacaster-type]</MediaCasterType>
  <StreamName>[stream-name]</StreamName>
</StartupStream>
```

Where:

- [application] is the name of live application that re-streams the source stream
- [mediacaster-type] is a valid mediacaster type: rtp, rtp-record, shoutcast, shoutcast-record, liverepeater, and
- [stream-name] is the name of the source stream

For example, to create a stream entry in **StartupStreams.xml**, do the following:

1. Click the **Server** tab in Wowza Streaming Engine Manager and then click **Startup Streams** in the contents panel.
2. On the **Virtual Host Startup Streams** page, click **Add Startup Stream**.
3. In the **Add to Startup Streams** dialog box, configure the options to create the entry in the **StartupStreams.xml** file and then click **OK**.

For more information, see [Start streams when Wowza Streaming Engine starts](#).

Note

Two server-side methods can also be used to start and stop streams using the MediaCaster system: **IApplicationInstance.startMediaCasterStream(...)**; and **IApplicationInstance.stopMediaCasterStream(...)**; For more information about these methods, see the [Wowza Streaming Engine Java API reference documentation](#).

Live stream repeater (origin/edge live streaming)

A live stream repeater uses multiple Wowza Streaming Engine servers in an origin/edge configuration to deliver live content across multiple servers. The encoded stream is ingested by the Wowza Streaming Engine origin server in the same way as any other single incoming source stream. Separately, a player requests the stream from an edge server, which maintains a single connection per-unique stream to the origin. Origin/edge configuration occurs at the application level. A single Wowza Streaming Engine instance can be configured as an origin for one application and as an edge for another application.

Here's an example that uses a single origin server with an application named **liveorigin**. To configure the origin server, do the following:

1. In Wowza Streaming Engine Manager, click the **Applications** tab.
2. On the **Add Application** page, click **Live**.
3. In the **New Application** dialog box, enter the following application name: **liveorigin**
4. On the **liveorigin** application page, select the following **Playback Types**:
 - **MPEG-DASH**
 - **Apple HLS**
 - **Adobe HDS**
 - **Microsoft Smooth Streaming**
5. Click **Save**.

To configure an edge server, do the following (repeat on each edge server):

1. In Wowza Streaming Engine Manager, click the **Applications** tab.
2. On the **Add Application** page, click **Live Edge**.
3. In the **New Application** dialog box, enter the following application name: **liveedge**
4. On the **liveedge** application page, select the following **Playback Types**:
 - **MPEG-DASH**
 - **Apple HLS**
 - **Adobe HDS**
 - **Microsoft Smooth Streaming**
5. If low latency is important, select the **Low-latency stream** check box (this adds extra load to the server).
6. In **Primary Origin URL**, Enter the URL of the **liveorigin** application using the WOWZ protocol URL prefix (wowz://). For example, if the origin server uses the domain name **origin.mycompany.com**, the value would be:
`wowz://origin.mycompany.com/liveorigin`
7. Click **Save**.

In the following examples, assume that the origin server uses the domain name **origin.mycompany.com** and that there are three edge servers with the domain names **edge1.mycompany.com**, **edge2.mycompany.com**, and **edge3.mycompany.com**. If the stream name is **mycoolevent**, the URLs for players streaming from **edge1** would be:

HLS (when Cupertino or CMAF streaming is enabled)

```
http://edge1.mycompany.com:1935/liveedge/mycoolevent/playlist.m3u8
```

MPEG-DASH

```
http://edge1.mycompany.com:1935/liveedge/mycoolevent/manifest.mpd
```

HDS

```
http://edge1.mycompany.com:1935/liveedge/mycoolevent/manifest.f4m
```

Smooth Streaming

```
http://edge1.mycompany.com:1935/liveedge/mycoolevent/Manifest
```

You can configure more than one origin server to provide a hot backup if the primary origin server goes offline. For example, if the failover origin server has the domain name **origin2.mycompany.com**, and it's configured identically as the primary origin server, you would specify the following **Secondary Origin URL** value in the **liveedge** application page on each edge server:

```
wowz://origin2.mycompany.com/liveorigin
```

Edge servers try to connect to the first origin server, and if this fails, they try to connect to the second origin server.

This example assumes that you're using an encoder in which the stream name is a simple name and not a URL. If you're using an encoder such as an MPEG-TS encoder in which the stream name isn't a simple stream name, you can use .stream files on the origin server to hide the complex stream names. For example, if your complex stream name on the origin server is **udp://0.0.0.0:10000**, use Stream files in Wowza Streaming Engine Manager to create a file named **mycoolevent.stream** and set the contents to **udp://0.0.0.0:10000**. You can then use **mycoolevent.stream** in place of **mycoolevent** in the example URLs above to play the stream.

Notes

- WOWZ is a TCP-based messaging protocol in Wowza Streaming Engine and is used for server-to-server communication. It's enabled by default. If one of the servers in the origin/edge configuration is running a version of Wowza Streaming Engine that doesn't support WOWZ, an RTMP connection is established instead.

- You can secure the connection between Wowza Streaming Engine servers in and origin/edge configuration by using a SecureToken shared secret. See [Configure a live stream repeater in Wowza Streaming Engine](#).
- If you use a non-push-based encoder (native RTP or MPEG-TS) and streaming players using any of the HTTP streaming protocols, you must use Startup Streams in Wowza Streaming Engine Manager to start the stream on the origin server and keep it running. Streams don't need to be kept running on edge servers.
- To provide load balancing between edge servers, you can use the dynamic load balancing system. See [Get the Dynamic Load Balancing AddOn for Wowza Streaming Engine](#).

Live stream recording

There are multiple ways to record live streams to VOD files for later playback, but using **Incoming Streams** for live applications in Wowza Streaming Engine Manager gives you the most control over the recording process. You can split in-process live stream recording archives into multiple on demand MP4 (QuickTime container) or FLV (Flash Video container) files automatically, with the split points based on video duration, clock time, or file size. The user interface shows all current live source streams and enables you to control when the recording starts and stops, the file name and locations, the container format, and other details. You can also control the live stream recording process using HTTP URL queries and programmatically using the **LiveStreamRecordManager** APIs. See [Record live streams in Wowza Streaming Engine](#).

For the **Live** and **Live HTTP Origin** application types in Wowza Streaming Engine Manager, you can select the **Record all incoming streams** option to record all streams published to the application by a live source. This recording option uses the **live-record** stream type and creates a recording with a file name that's the same as the source stream name in the application's streaming file directory. To stop recording all source streams to these application types, clear the **Record all incoming streams** option and restart the application.

Finally, you can record IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, and streaming output from native RTP or MPEG-TS encoders using the MediaCaster system. Stream files and Startup Streams in Wowza Streaming Engine Manager use the MediaCaster system to pull a stream from a stream source and make it available for streaming to all player technologies supported by Wowza Streaming Engine. You can configure these features to record the source streams instead by selecting an appropriate ***-record** stream type for the MediaCaster type (such as **rtp-record** for IP camera streams) and the streams are recorded to the streaming file directory for the selected application. See [MediaCasters, Stream files, and Startup Streams](#).

Notes

- The ***-record** stream types are the easiest to use but provide the least amount of control. If you use this method, the entire duration of the published stream is recorded to a single file in the live application's streaming file directory. If the stream source starts and stops, the file is versioned with a version number and a new file is started. You can control the container format used (MP4 or FLV) by specifying a stream name prefix in the stream source. If you specify the mp4: prefix, the stream is recorded to an MP4 (QuickTime) container. An MP4 container can only record H.264, AAC, and MP3 media data. If you specify the flv: prefix, the stream is recorded to an FLV container. The FLV container is the only option if you're recording with Flash Player.
- If you use one of the ***-record** stream types and also configure Incoming Streams for a live application to record a live source stream, two or more copies of the recording are created in the live application's streaming file directory by default. The ***-record** stream types record the stream to a single file and the recorded file name is the same as the stream name. The Incoming Streams feature creates one or more recordings with file names that include the stream name and other information, depending on selected segmentation and versioning options.
- The WebcamRecording example in the Wowza Streaming Engine installation is a specialized way to record a remote live stream when using Adobe Flash Player. It uses the **record** stream type and built-in Flash Player capabilities to control the recording process.

Virtual hosting

Wowza Streaming Engine can be configured to run multiple virtual host (VHost) environments on a single server. This lets multiple users share a server in separate environments. Each VHost environment has its own set of configuration files, application folders, and log files and can be configured with its own system resource and streaming limitations. By default, Wowza Streaming Engine is configured with a single VHost named **_defaultVHost_**.

Configuration files

The **VHosts.xml** configuration file in the Wowza Streaming Engine **[install-dir]/conf** folder defines each VHost environment. The following items are required in **VHosts.xml** to define a VHost:

- **VHosts/VHost/Name** – The name of the VHost
- **VHosts/VHost/ConfigDir** – The configuration directory for the VHost
- **VHosts/VHost/ConnectionLimit** – The maximum number of simultaneous connections that the VHost supports. If this value is **0**, the VHost can support an unlimited number of simultaneous connections.

Typical configuration

A typical **VHosts.xml** file for a VHost environment contains two VHosts. The following example shows the default VHost (**_defaultVHost_**) and a new VHost (**_newVHost_**):

```
<Root>
  <VHosts>
    <VHost>
      <Name>_defaultVHost_</Name>
      <ConfigDir>${com.wowza.wms.ConfigHome}</ConfigDir>
      <ConnectionLimit>0</ConnectionLimit>
    </VHost>
    <VHost>
      <Name>_newVHost_</Name>
      <ConfigDir>${com.wowza.wms.ConfigHome}/newVHost</ConfigDir>
      <ConnectionLimit>0</ConnectionLimit>
    </VHost>
  </VHosts>
</Root>
```

The directory structure for the VHosts in the above example would be:

```
[install-dir]
  [defaultVHost]
    [applications]
    [conf]
      Application.xml
      clientaccesspolicy.xml
      crossdomain.xml
      MediaCache.xml
      StartupStreams.xml
      Tune.xml
      VHost.xml
      admin.password
      publish.password
    [content]
    [keys]
    [logs]
    [transcoder]
  [newVHost]
    [applications]
    [conf]
      Application.xml
      clientaccesspolicy.xml
      crossdomain.xml
      MediaCache.xml
      StartupStreams.xml
      Tune.xml
      VHost.xml
      admin.password
      publish.password (Optional, see Notes below)
    [content]
    [keys]
    [logs]
    [transcoder]
```


Notes

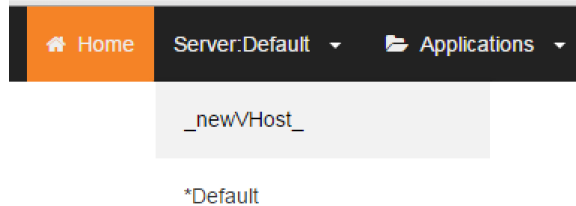
- By default, all VHost environments share the **publish.password** file for the default VHost. You can use source authentication in Wowza Streaming Engine Manager to set up unique publishing credentials for each VHost and the unique credentials are stored in this file.

Alternatively, you can retain the **publish.password** file when you copy the **[install-dir]/conf** folder to your new VHost environment and then configure the **securityPublishPasswordFile** property for new VHost applications to reference this file for publishing credentials. If you do this, you can't use source authentication in Wowza Streaming Engine Manager to update the file. See [Configure security using Wowza Streaming Engine Manager](#).

- For more information about how to configure per-VHost logging, see [Logging](#).

VHosts are defined in the **VHosts.xml** file located at **[install-dir]/conf/**. Each VHost gets its own configuration directory structure with its own set of configuration files and **application**, **conf**, and **logs** folders. VHosts can be added, modified, and deleted through the **VHosts.xml** configuration file. If you change **VHosts.xml** while Wowza Streaming Engine is running, the changes take effect after restarting the server.

After adding a new VHost to **VHosts.xml** and creating its directory structure, select the new VHost on the **Server** tab in Wowza Streaming Engine Manager to manage it.



Wowza Streaming Engine only supports IP address/port-based virtual hosting. It doesn't support domain name-based virtual hosting. In **VHost.xml**, each VHost must define **HostPort** entries with unique IP address and port combinations that don't conflict with other VHosts that are defined on the server. The following combinations represent valid VHost port configurations:

```
defaultVHost:
<HostPort>
  <IpAddress>192.168.1.2</IpAddress>
  <Port>1935</Port>
</HostPort>

newVHost:
<HostPort>
  <IpAddress>192.168.1.2</IpAddress>
  <Port>1936</Port>
</HostPort>
```

-or-

```
defaultVHost:
<HostPort>
  <IpAddress>192.168.1.2</IpAddress>
```

```
<Port>1935</Port>
<HostPort>

newVHost:
<HostPort>
  <IpAddress>192.168.1.3</IpAddress>
  <Port>1935</Port>
<HostPort>
```

To set up the IP address and port values, click the **Server** tab in Wowza Streaming Engine Manager, select a VHost in the list, and then click **Virtual Host Setup** in the contents panel. In the **Virtual Host Setup** page, click **Edit** to update the IP addresses and port values for the default host ports.

Publishing with the Stream and Publisher classes

Wowza Streaming Engine includes the **Stream** class and the **Publisher** class for doing server-side publishing. The **Stream** class is a high-level server-side API for mixing live and VOD content on the fly into a single destination stream and lets you do television-style publishing. It also includes a package that enables creation of a server-side XML-based playlist. For more information about the **Stream** class, see [Schedule streaming with Wowza Streaming Engine \(StreamPublisher\)](#).

The **Publisher** class is a low-level publishing API that lets you inject raw compressed video and audio frames into Wowza Streaming Engine to create a custom live stream. See the **Publisher** class in the [Wowza Streaming Engine Java API reference documentation](#) for details.

Using Wowza Streaming Engine Manager

Wowza Streaming Engine Manager lets you set up, manage, and monitor streams using a web browser. It extends the programmatic and command line configuration and management of Wowza Streaming Engine, enabling publishers with a diverse range of technical abilities to have greater control and confidence when streaming video.

You can use Wowza Streaming Engine Manager with the latest versions of most modern web browsers that support HTML5 and CSS 3. We recommend that you use the Google Chrome browser.

Starting and stopping Wowza Streaming Engine Manager

Notes

- Wowza Streaming Engine must be started to use Wowza Streaming Engine Manager. See [Starting and stopping the software](#).
- Wowza Streaming Engine Manager can't run as a service and in standalone mode at the same time.
- After starting Wowza Streaming Engine Manager, open it in a web browser with the URL **`http://[wowza-ip-address]:8088/enginemanager`**, where **`[wowza-ip-address]`** is the Wowza Streaming Engine IP address or domain name.
- For more information about how to sign in to Wowza Streaming Engine Manager, see [Managing sign-in credentials](#).

Start and stop Wowza Streaming Engine Manager as a service (Windows)

To start the service:

1. Press WIN key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine Manager 4.7.8**, and then click **Start**.

To stop the service:

1. Press WIN key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine Manager 4.7.8**, and then click **Stop**.

You can set Wowza Streaming Engine Manager to start automatically as a Windows service when Windows starts. To stop the service from starting automatically when Windows starts:

1. Press WIN key + R, type **services.msc** in the **Run** dialog box, and then click **OK**.
2. In the **Services** window, right-click **Wowza Streaming Engine Manager 4.7.8**, and then click **Properties**.
3. In the **Properties** dialog box, on the **General** tab, set **Startup type** to **Manual**.

Start and stop Wowza Streaming Engine Manager in standalone mode (Windows)

To start Wowza Streaming Engine Manager in standalone mode, make sure that the Wowza Streaming Engine Manager service is stopped (see above), and then do the following:

1. Press WIN key + R, type **cmd** in the **Run** dialog box, and then press Enter.
2. In the Command Prompt window, execute the following commands:

```
cd %WMSAPP_HOME%\manager\bin
startmgr.bat
```

To stop Wowza Streaming Engine Manager:

1. Press WIN key + R, type **cmd** in the **Run** dialog box, and then press Enter.
2. In the Command Prompt window, execute the following commands:

```
cd %WMSAPP_HOME%\manager\bin
shutdownmgr.bat
```

Start and stop Wowza Streaming Engine Manager as a service (macOS)

To start the service, double-click the **Start Services** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following command:

```
sudo launchctl load -w  
/Library/LaunchDaemons/com.wowza.WowzaStreamingEngineManager.plist
```

To stop the service, double-click the **Stop Services** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following command:

```
sudo launchctl unload -w  
/Library/LaunchDaemons/com.wowza.WowzaStreamingEngineManager.plist
```

Note

The **Start Services** and **Stop Services** applications also start and stop the Wowza Streaming Engine system service. See [Starting and stopping the software](#).

Start and stop Wowza Streaming Engine Manager in standalone mode (macOS)

To start Wowza Streaming Engine in standalone mode, double-click the **Start Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.7.8/manager/bin  
./startmgr.sh
```

To stop the manager, double-click the **Stop Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.7.8** or open a Terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.7.8/manager/bin  
./shutdownmgr.sh
```

Note

The **Start Standalone Mode** and **Stop Standalone Mode** applications also start and stop Wowza Streaming Engine in standalone mode. See [Starting and stopping the software](#).

Start and stop Wowza Streaming Engine Manager as a service (Linux)

Note

The operations in this section must be performed as the **root** user with **sudo** access.

To start the service, enter one of the following commands, depending on your Linux distribution:

```
sudo service WowzaStreamingEngineManager start
```

-or-

```
/etc/init.d/WowzaStreamingEngineManager start
```

To stop Wowza Streaming Engine Manager, enter:

```
sudo service WowzaStreamingEngineManager stop
```

-or-

```
/etc/init.d/WowzaStreamingEngineManager stop
```

Note

If these instructions don't apply to your Linux distribution, consult your Linux manual.

Start and stop Wowza Streaming Engine Manager in standalone mode (Linux)

To start the manager in standalone mode, open a Terminal window and enter the following commands:

```
cd /usr/local/WowzaStreamingEngine/manager/bin  
./startmgr.sh
```

To stop Wowza Streaming Engine Manager, enter:

```
cd /usr/local/WowzaStreamingEngine/manager/bin  
./shutdownmgr.sh
```

Managing sign-in credentials

The first time you start Wowza Streaming Engine Manager, you'll be asked to sign in with the case-sensitive user name and password that you created when you installed Wowza Streaming Engine. This account has administrator access to enable control of the Wowza Streaming Engine server through the web-based manager UI. However, it doesn't by default provide access to advanced property, server listener, and module settings, which are reserved for expert Wowza Streaming Engine users.

After you sign in, you can enable access to the advanced settings for the default administrator account and add accounts for other users. You can create additional user accounts with both administrative and read-only access.

Enable access to advanced settings for the default administrator account

1. In Wowza Streaming Engine Manager, click the **Server** tab and then click **Users** in the contents panel.
2. On the **Users** page, click the user name for the administrator account in the **Users** list.
3. Click **Edit**, and then select the **Allow access to advanced properties and features** check box.
4. (Optional) Enter a new password in the **Password** and **Confirm Password** fields. The password values are case-sensitive.
5. Click **Save**. You'll be signed out and must sign in again.

You can also enable access to the advanced settings for the default administrator account by updating the `[install-dir]/conf/admin.password` file using a text editor. For example, to specify that the **Admin** user can access the advanced settings, specify the **advUser** group as shown in the following example:

```
# Admin password file (format [username] [space] [password] [space] [group])
#username password group|group
Admin AdminPassword admin|advUser
```

Administrators can create accounts for other users with full administrative access to Wowza Streaming Engine Manager or with read-only privileges.

Create user accounts

1. In Wowza Streaming Engine Manager, click the **Server** tab and then click **Users** in the contents panel.
2. On the **Users** page, click **Add User**.
3. Enter a name for the user in **User Name** and a password for the user in **Password** and **Confirm Password** fields. The user name and password values are case-sensitive.
4. Specify the access level (**Read-Only** or **Administrator**) for the new user by selecting the appropriate **Access Level** option.
5. To enable the new user to either manage (**Administrator** user) or view (**Read-Only** user) advanced settings, select the **Allow access to advanced properties and features** check box.
6. Click **Add**.

You can also add new user accounts by updating the `[install-dir]/conf/admin.password` file using a text editor. For example, to add the **newAdmin** and **readOnly** user accounts with access to advanced settings, edit the **admin.password** file as follows.

```
# Admin password file (format [username] [space] [password] [space] [group])
#username password group|group
Admin AdminPassword admin|advUser

newAdmin newAdminPassword admin|advUser
readOnly readOnlyPassword advUser
```

The **readOnly** user can view the advanced settings but can't change them.

Navigating Wowza Streaming Engine Manager

Here's a tour of the Wowza Streaming Engine Manager user interface. For more information, see [Find your way around Wowza Streaming Engine Manager](#).

Home page

The screenshot shows the Wowza Streaming Engine Manager interface. At the top is a navigation bar with tabs for Home, Server, and Applications, and links for myUser, Help, and Sign Out. Below the navigation bar, the main content area is divided into several sections. On the left, there's a 'Welcome to Wowza Streaming Engine!' message, followed by 'Monthly Subscription', 'Status' (with a 'Connections' chart and 'Usage' chart), 'Server Uptime', and 'Test Video'. On the right, there's a 'Performance Warning!' box, 'Application Connection Settings' (with a table for Host, Port, Application, Stream Name, and Login), and 'Getting Started With Applications'. Numbered callouts (1-5) point to specific elements: 1 points to the navigation bar, 2 points to the Status section, 3 points to the Application Connection Settings table, 4 points to the Test Video button, and 5 points to the Getting Started With Applications section.

- 1 Click the tabs on the menu bar to access features that help you manage the server and virtual host (the **Server** tab) and create and manage live and video-on-demand application types (the **Applications** tab). Click the **Help** link to access articles and resources on the Wowza website that help you configure streaming workflows.
- 2 View information in the **Status** area about how the total number of connections (both source and playback connections) for the server (the **Connections** chart) and the total server resource consumption for CPU, Java heap, memory, and disk (the **Usage** chart). You can also see if Transcoder, nDVR, and DRM are licensed and enabled, and which applications they're enabled for.

- 3 Use the information (IP address and port) shown in **Application Connection Settings** to publish a stream to the server from your encoder or camera.
- 4 Quickly verify that the server is up and running by using built-in test players to stream the **sample.mp4** video file that's installed with the server software.
- 5 Use the **Getting Started** information to quickly jump to configuration areas in Wowza Streaming Engine Manager and to get more information about the Support resources that are available if you have problems.

Server configuration

The screenshot shows the Wowza Streaming Engine Manager interface. At the top is a navigation bar with 'Home', 'Server', and 'Applications' tabs. Below this is a sidebar (labeled 1) with a 'SERVER' section containing 'Server Setup' (highlighted), 'Server Monitoring', 'Virtual Host Setup', 'Virtual Host Monitoring', 'Transcoder', 'Media Cache', 'Users', 'Source Authentication', 'Performance Tuning', 'Logs', and 'About'. The main content area (labeled 2) is titled 'Server Setup' and contains the following information:

- Name:** Wowza Streaming Engine
- Description:** Wowza Streaming Engine is robust, customizable, and scalable server software that powers reliable streaming of high-quality video and audio to any device, anywhere.
- License Keys:** ENGM4-XXXXX-XXXXX-XXXXX-VJ6rV
- Default Stream Prefix:** mp4
- Options:**
 - ☒ Enable monitoring and statistics for this server
 - ☒ Allow RTP datagram port sharing
- RTP Datagram Starting Port:** 6970

In the top right corner of the 'Server Setup' section is a 'Restart...' button (labeled 3). In the middle right of the 'Server Setup' section is a '< Show Help' button (labeled 4).

- 1 The contents panel provides access to the following features that let you configure, manage, and monitor the server and virtual hosts (VHosts).
 - Server Setup** – Configure settings such as the Wowza Streaming Engine instance name, available license keys, and enable and disable Monitoring for the server and its applications.
 - Server Monitoring** – Monitor server resource consumption (CPU, memory, Java heap, and disk usage), source and playback connections, network throughput, and uptime. See [Monitor server connections, load, and application statistics in Wowza Streaming Engine](#).
 - Virtual Host Setup** – Manage virtual hosting environments. By default, Wowza Streaming Engine ships with a single VHost environment named `_defaultVHost_`, however, you can add more VHost environments and manage them separately. See [Virtual hosting](#).

Virtual Host Monitoring – Monitor VHost source and playback connections, network throughput, and uptime. See [Monitor server connections, load, and application statistics in Wowza Streaming Engine](#).

Transcoder – Monitor the number of concurrent live source streams ingested by the Transcoder and add, modify, and delete Transcoder templates. See [Use Wowza Streaming Engine Transcoder](#).

Media Cache – Configure the read-through caching mechanism that enables scaling of VOD streams by re-streaming VOD file sources from HTTP-based servers that support HTTP/1.1 range requests and from network-attached file systems. See [Scale video-on-demand streaming with Wowza Streaming Engine Media Cache](#).

Users – Manage administrator and user accounts for Wowza Streaming Engine Manager. See [Managing sign-in credentials](#).

Source Authentication – Manage user names and passwords that RTMP-based and RTSP-based encoders and cameras can use to connect and publish a live stream if the live application requires authentication.

Performance Tuning – Adjust performance settings from the default values used when the server starts. See [Tune Wowza Streaming Engine for optimal performance](#).

Logs – View Wowza Streaming Engine and Wowza Streaming Engine Manager log files. Filtering and display options let you customize what you see in the UI, and you can download large log files to a compressed (zipped) folder for offline viewing. See [View log messages in Wowza Streaming Engine Manager](#).

About – View information about Wowza Streaming Engine such as the installed version number, license details, and the version of Java being used.

Startup Streams – Pull live IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, and streams from native RTP or MPEG-TS encoders and start them automatically when the VHost starts. See [Startup Streams](#).

Stream Files – Replace (alias) complex stream names that are published to Wowza Streaming Engine from sources such as IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, and streams from native RTP or MPEG-TS encoders. See [Stream files](#).

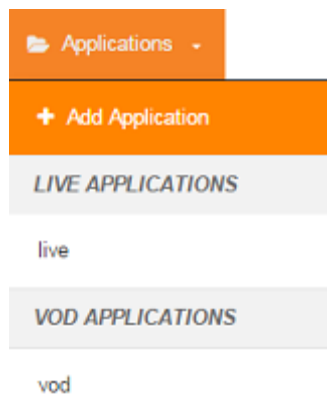
SMIL Files – Create Synchronized Multimedia Integration Language (SMIL) files that organize streams of various bitrates into groups for HTTP adaptive bitrate streaming. See [Stream adaptive-bitrate content in Wowza Streaming Engine](#).

- 2 When you click a feature in the contents panel, a page displays that enables you to configure the feature's settings. Advanced settings for fine-tuning the server configuration are available for some server features on **Properties** and **Server Listeners** tabs. These tabs are only available to users with advanced permissions. See [Advanced properties and settings](#).

- 3 Some features have buttons in the upper-right corner that provide additional functionality. Some server-level features let you restart the server and stop and restart the VHost.
- 4 The Help panel provides details about how to configure the controls on the feature page. Toggle the panel's visibility by clicking **Hide Help** or **Show Help**.

Application types

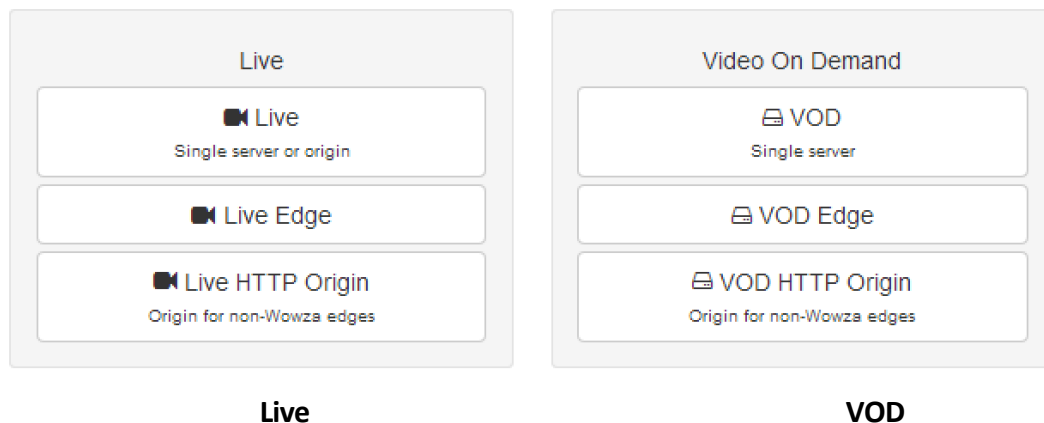
An *application* is a set of configuration options in Wowza Streaming Engine that supports a specific workflow for the delivery of streaming content. To add applications in Wowza Streaming Engine Manager, click the **Applications** tab and then click **Add Application**.



In the **Add Application** page that appears, you can add applications for six streaming scenarios.

Add Application

Select an Application Type.



Delivers live streams to players (single server) or as an origin server to deliver live streams to other servers running Wowza Streaming

Delivers video-on-demand streams to players (single server).

Engine to scale content delivery to a large number of players.

Live Edge

Ingests live streams from a live application on an origin server that's running Wowza Streaming Engine, then delivers the live streams to players (single server).

VOD Edge

Ingests video-on-demand files from a Media Cache source and streams them to players (single server).

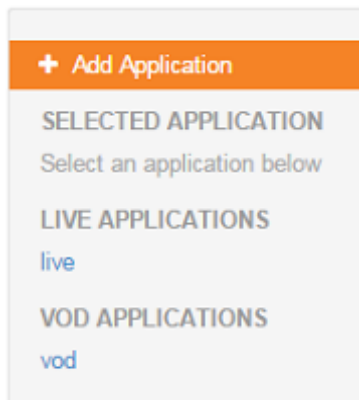
Live HTTP Origin

Delivers live streams to an HTTP caching infrastructure using HTTP streaming protocols (HLS, MPEG-DASH, HDS, and Smooth Streaming).

VOD HTTP Origin

Delivers video-on-demand files to an HTTP caching infrastructure using HTTP streaming protocols (HLS, MPEG-DASH, HDS, and Smooth Streaming).

To add an application, click the **Application Type** in the page that corresponds to your needs, enter a name for the application in the **New Application** dialog box, and then click **Add**. Single instances of a live application type (named **live**) and an on-demand application type (named **vod**) are included in the default installation of Wowza Streaming Engine.



Application configuration

The screenshot shows the Wowza Streaming Engine Manager interface. The top navigation bar includes links for Home, Server, Applications, myUser, Help, and Sign Out. The sidebar on the left lists various features under 'SELECTED APPLICATION' and 'LIVE APPLICATIONS'. The main content area displays the configuration for the 'live' application, including playback types, options, streaming file directory, closed caption sources, and maximum connections. Numbered callouts (1-4) highlight specific features: 1 points to the 'Add Application' button, 2 points to the 'Edit' button, 3 points to the 'Test Players...' button, and 4 points to the 'Show Help' button.

- 1 The contents panel provides access to the following features that let you configure, manage, and monitor application types.

Application Setup – Modify application settings such as the playback types (HTTP streamers and packetizers), default content storage location, closed-captioning options, and other settings. Some settings vary by application type. All application types

Monitoring – Monitor application source and playback connections, network throughput, and uptime. See [Monitor server connections, load, and application statistics in Wowza Streaming Engine](#). All application types

Sources (Live) – Get connection information for encoders and cameras that publish a stream to this application. If you're viewing this page on your iOS or Android mobile device that has the Wowza GoCoder™ encoding app installed, you can automatically configure the GoCoder app to publish a stream to this application. Wowza Streaming Engine supports integrated configuration of additional live sources provided by Works With Wowza partners. You can All live application types

connect many other live sources to live Wowza Streaming Engine applications, but you must configure the connection settings manually. See [Connect a live source to Wowza Streaming Engine](#).

Stream Files – Replace (alias) complex stream names that are published to the application from sources such as IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, and streams from native RTP or MPEG-TS encoders. See [Stream files](#).

Live
Live Edge

Incoming Streams – View details about live streams that are published to this application and record them to VOD files for later playback. See [Record live streams in Wowza Streaming Engine](#).

All live application types

Wowza Player – Send live or on-demand HLS stream URLs to Wowza Player Builder, and then use Player Builder settings to customize your player and generate player embed code for your webpage or web application. See [Get started with Wowza Player](#).

Live
VOD
VOD Edge

Stream Targets – Send live streams to CDNs, streaming servers, streaming services, and multicast networks for distributed delivery. See [Distribute live streams](#).

All live application types

Source Security – Configure options for securing RTMP and RTSP-based source connections to this application (for example, from RTMP-based encoders).

All live application types

Playback Security – Configure options for securing playback connections to Wowza Streaming Engine. For example, require a secure RTMP connection, specify a security token ("shared secret"), and restrict playback to specific IP addresses.

All application types

SMIL Files – Create Synchronized Multimedia Integration Language (SMIL) files that organize streams of various bitrates into groups for HTTP adaptive bitrate streaming. See [Stream adaptive-bitrate content with Wowza Streaming Engine](#).

Live
Live Edge
VOD
VOD Edge

nDVR – Configure DVR playback of live streams. See [nDVR](#).

All live application types

Transcoder – Configure transcoding of live streams to suit desired playback devices. See [Use Wowza Streaming Engine Transcoder](#).

Live
Live HTTP Origin

DRM – Integrate with Key Management Service partners to enable on-the-fly DRM encryption of premium live and

Live
Live Edge

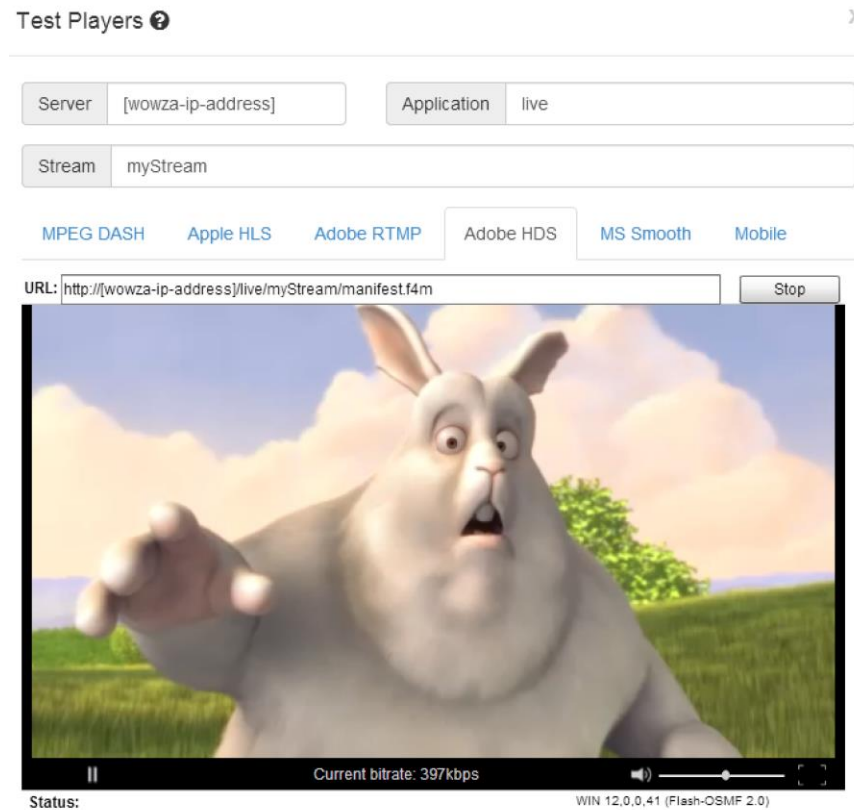
VOD content for a variety of playback devices. See [Stream encryption with DRM](#). VOD
VOD Edge

- 2 When you click an application or one of its features in the contents panel, a page displays that enables you to configure the application or feature settings. Advanced settings for fine-tuning the configuration are available for the application and some application features on **Properties** and **Modules** tabs. These tabs are only available to users with advanced permissions. See [Advanced properties and settings](#).
- 3 Most application and feature pages have buttons in the upper-right corner that provide additional functionality. You can access test players to test your streams, copy application settings to create a new application with identical settings, restart an application, and delete an application.
- 4 The **Help** panel provides details about how to configure the controls on the application or feature page. Toggle the panel's visibility by clicking **Hide Help** or **Show Help**.

Test players

Wowza Streaming Engine Manager provides test players for all of the live and on-demand application types so that you can test your streaming configurations. To access the test players, click the **Test Players** button in the upper-right corner of the application or feature page. Then in the **Test Players** dialog box, click the tab for the protocol you want to test.

The test players for live applications are preconfigured to play a stream named **myStream** from the default **live** application on the local Wowza Streaming Engine instance. If you configured your encoder or camera to publish a stream to the **live** application with a different stream name, be sure to substitute it in place of **myStream** in the **Stream** box.



The test players for VOD applications are preconfigured to play the **[install-dir]/content/sample.mp4** video file that installs with Wowza Streaming Engine.


Test Players ? X

Server Application

Media File Name

[MPEG DASH](#)
[Apple HLS](#)
[Adobe RTMP](#)
[Adobe HDS](#)
[MS Smooth](#)
[Mobile](#)

URL: Stop



Status: Playing WIN 12.0.0.41 (Flash-OSMF 2.0)

To use your own VOD file, copy it to the **[install-dir]/content** root folder and substitute its file name for **sample.mp4** in the **Media File Name** field. If your custom VOD file isn't stored in the **[install-dir]/content** root folder, you must add the default application instance name to the playback URL. For example, if the **sample.mp4** video file is in **[install-dir]/content/myVideos**, enter **vod/_definst_** in the **Application** field and enter **mp4:/myVideos/sample.mp4** in the **Media File Name** field.

Note

The test players can't display closed captions or play encrypted streams. DVR playback is only supported by the HDS, HLS, and Microsoft Smooth Streaming test players. If you change the default stream values to playback a new stream, you may need to restart the test players.

Advanced properties and settings

Advanced settings for fine-tuning the server and application configuration are available in a couple of locations in Wowza Streaming Engine Manager. Some server features have advanced settings on **Properties** and **Server Listener** tabs to adjust the server configuration, while applications and some application features have **Properties** and **Modules** tabs to adjust the application configuration. The tabs that provide access to the advanced properties and settings aren't visible unless the signed-in user has advanced permissions. Administrators with advanced permissions can configure the advanced properties and settings while read-only users can't change the advanced properties and settings. See [Managing sign-in credentials](#).

Properties pages may have many properties that you can configure, so they're organized into categories. Click a link in the **Quick Links** area at the top of the page to jump to the associated property settings. For example, click **Closed Captions**:

live

Live Single Server or Origin

Setup Properties Modules

Note: Items on this page should be configured by advanced users only.

Quick Links Use the following links to jump to the correct section on this page.

HTTP Streamers Cupertino Settings MediaCaster Stream Monitor RTP Jitter Buffer
RTSP/RTP Window Title StreamRecorder Defaults Streams **Closed Captions**
Cupertino Streaming Packetizer MPEG-DASH Streaming Packetizer
San Jose Streaming Packetizer Smooth Streaming Packetizer Custom

To jump to the **Closed Captions** property settings for an application:

Closed Captions Properties for tuning closed captioning functionality in streams and for enabling debug logging for specific components of the closed captioning implementation.

[Return to top ↑](#)

[Edit](#)

Enabled	Name	Value
	captionUndefinedLanguageId	
	cea608CaptionConverterColor	
	maximumCaptionDuration	
	closedCaptionLiveMaxDisplayTime	
	cclngestCEA608EnableField1	
	cclngestCEA608EnableField2	
	cclngestCEA608LogCaptions	

Many technical articles on the Wowza website prescribe custom properties for tuning the server and applications and to add advanced functionality. Each article describes how to add the custom properties using the **Custom Properties** area on a **Property** tab:

Custom Custom properties added by you to extend Wowza Streaming Engine functionality.

[Return to top ↑](#)

[Edit](#)

No custom properties defined.

If you can't find the property that you're looking for in the previous sections, click the **Edit** button and then click the **Add Custom Property** button on the **Custom Property** page.

See [Properties](#).

Server administration

Wowza Streaming Engine can run standalone from a command shell or as a system service. Running standalone is best for developing custom applications because the server can be started and stopped quickly, and server log messages can be viewed immediately in the console window. Running as a system service is more often used for server deployments where the server must continue to run after you log off the computer or must be automatically started when the computer is rebooted.

Configuring SSL and RTMPS

Wowza Streaming Engine supports Secure Sockets Layer (SSL)—RTMPS (RTMP over SSL) and HTTPS (HTTP over SSL)—streaming protection. SSL allows web browsers and web servers to communicate over a secure connection, with the encrypted data being sent and received in both directions. You can use Wowza StreamLock AddOn to get a free 256-bit SSL certificate, you can get an SSL certificate from a certificate authority, or you can create a certificate yourself (a self-signed SSL certificate).

Notes

- For information on getting an SSL certificate from Wowza, see [Get SSL certificates from the Wowza Streaming Engine StreamLock service](#).
- For information on getting an SSL certificate from a certificate authority, see [Request an SSL certificate for Wowza Streaming Engine from a certificate authority](#).
- For information on creating a self-signed SSL certificate, see [Create a self-signed SSL certificate for Wowza Streaming Engine](#).

Logging

Wowza Streaming Engine uses the Apache log4j logging utility as its logging implementation. The log4j logging system provides ample functionality for log formatting, log rolling, and log retrieval for most applications. By default, Wowza Streaming Engine is configured to log basic information to the server console and detailed information in the W3C Extended Common Log Format (ECLF) to a log file. Java messaging is also captured in the log files to help monitor and aid troubleshooting. The log files are written to the **[install-dir]/logs** folder.

For information about common log messages and suggestions for resolution, see [Troubleshoot error messages in Wowza Streaming Engine](#).

Logging fields

Wowza Streaming Engine can generate the following logging fields:

Field name	Description
c-client-id	Client ID number assigned by the server to the connection
c-ip	Client connection IP address
c-proto	Client connection protocol: http (HLS), http (Smooth Streaming), rtmp, rtmpe, rtmps (HTTP-1.1), rtmpt (HTTP-1.1), rtmpte (HTTP-1.1)
c-referrer	URL of the Flash movie that initiated the connection to the server
c-user-agent	Version of the Flash client that initiated the connection to the server
cs-bytes	Total number of bytes transferred from client to server (cumulative)
cs-stream-bytes	Total number of bytes transferred from client to server for stream x-stream-id (cumulative)
cs-uri-query	Query parameter for stream x-stream-id
cs-uri-stem	Full connection string for stream x-stream-id (excludes query parameters)
date	Date of log event
s-ip	IP address of the server that received this event
s-port	Port number through which the server received this event
s-uri	Full connection string on which the server received this event

sc-bytes	Total number of bytes transferred from server to client (cumulative)
sc-stream-bytes	Total number of bytes transferred from server to client for stream x-stream-id (cumulative)
time	Time of log event
tz	Time zone of log event
x-app	Name of the application from which the event was generated
x-appinst	Name of the application instance from which the event was generated
x-category	Log event category (server, vhost, application, session, stream)
x-comment	Extra comment about the log event
x-ctx	Extra data about the context of the log event
x-duration	Time, in seconds, that this event occurred within the lifetime of the x-category object
x-event	Log event (see Logging events)
x-file-ext	File extension of stream x-stream-id
x-file-length	File length, in seconds, of stream x-stream-id
x-file-name	Full file path of stream x-stream-id
x-file-size	File size, in bytes, of stream x-stream-id
x-severity	Log event severity (DEBUG, INFO, WARN, ERROR, FATAL)
x-sname	Name of stream x-stream-id
x-sname-query	Query parameters of stream x-stream-id
x-spos	Position, in milliseconds, within the media stream
x-status	Log event status (see Logging status values)
x-stream-id	Stream ID number assigned by the server to the stream object
x-suri	Full connection string for stream x-stream-id (includes query parameters)
x-suri-query	Query parameter for connection string
x-suri-stem	Full connection string for stream x-stream-id (excludes query parameters)
x-vhost	Name of the virtual host from which the event was generated

Logging events

Wowza Streaming Engine can generate the following logging events:

Event name	Description
announce	RTSP Session Description Protocol (SDP) ANNOUNCE
app-start	Application instance start
app-stop	Application instance shutdown
comment	Comment
connect	Connection result
connect-burst	Connection accepted in burst zone
connect-pending	Connection pending approval by application and license manager
create	Media or data stream created
decoder-audio-start	Audio decoding has started for a transcoded stream
decoder-audio-stop	Audio decoding has stopped for a transcoded stream
decoder-video-start	Video decoding has started for a transcoded stream
decoder-video-stop	Video decoding has stopped for a transcoded stream
destroy	Media or data stream destroyed
disconnect	Client (session) disconnected from server
encoder-audio-start	Audio encoding has started for a transcoded stream
encoder-audio-stop	Audio encoding has stopped for a transcoded stream
encoder-video-start	Video encoding has started for a transcoded stream
encoder-video-stop	Video encoding has stopped for a transcoded stream
pause	Playback has paused
play	Playback has started
publish	Start stream publishing
record	Start stream recording
recordstop	Stop stream recording
seek	Seek has occurred
setbuffertime	Client call to NetStream.setBufferTime(secs) logged in milliseconds

setstreamtype	Client call to <code>netConnection.call("setStreamType", null, "[streamtype]")</code>
server-start	Server start
server-stop	Server shutdown
stop	Playback has stopped
unpause	Playback has resumed from pause
unpublish	Stop stream publishing
vhost-start	Virtual host start
vhost-stop	Virtual host shutdown

Logging status values

Wowza Streaming Engine can generate the following logging status values:

Status value	Description
100	Pending or waiting (for approval)
200	Success
302	Rejected by application with redirect information
400	Bad request
401	Rejected by application
413	Rejected by license manager
500	Internal error

Logging configuration

Logging is configured in the **conf/log4j.properties** properties file. The log4j logging system has many logging configuration options. This section covers the basic options for enabling and disabling different logging fields, events, and categories.

The following example shows a basic **log4j.properties** file for a Wowza Streaming Engine instance:

```
log4j.rootCategory=INFO, stdout, serverAccess, serverError

# Console appender
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=com.wowza.wms.logging.ECLFPatternLayout
log4j.appender.stdout.layout.Fields=x-severity,x-category,x-event,x-ctx,x-
comment
log4j.appender.stdout.layout.OutputHeader=false
log4j.appender.stdout.layoutQuoteFields=false
```



```

log4j.appender.stdout.layout.Delimiter=space

# Access appender
log4j.appender.serverAccess=org.apache.log4j.WowzaDailyRollingFileAppender
log4j.appender.serverAccess.DatePattern='.'yyyy-MM-dd
log4j.appender.serverAccess.File=${com.wowza.wms.ConfigHome}/logs/wowzastre
amingengine_access.log
log4j.appender.serverAccess.layout=com.wowza.wms.logging.ECLFPatternLayout
log4j.appender.serverAccess.layout.Fields=x-severity,x-category,x-
event;date,time,c-client-id,c-ip,c-port,cs-bytes,sc-bytes,x-duration,x-
sname,x-stream-id,sc-stream-bytes,cs-stream-bytes,x-file-size,x-file-
length,x-ctx,x-comment
log4j.appender.serverAccess.layout.OutputHeader=true
log4j.appender.serverAccess.layout QuoteFields=false
log4j.appender.serverAccess.layout.Delimiter=tab

# Error appender
log4j.appender.serverError=org.apache.log4j.WowzaDailyRollingFileAppender
log4j.appender.serverError.DatePattern='.'yyyy-MM-dd
log4j.appender.serverError.File=${com.wowza.wms.ConfigHome}/logs/wowzastre
amingengine_error.log
log4j.appender.serverError.layout=com.wowza.wms.logging.ECLFPatternLayout
log4j.appender.serverError.layout.Fields=x-severity,x-category,x-
event;date,time,c-client-id,c-ip,c-port,cs-bytes,sc-bytes,x-duration,x-
sname,x-stream-id,sc-stream-bytes,cs-stream-bytes,x-file-size,x-file-
length,x-ctx,x-comment
log4j.appender.serverError.layout.OutputHeader=true
log4j.appender.serverError.layout QuoteFields=false
log4j.appender.serverError.layout.Delimiter=tab
log4j.appender.serverError.Threshold=WARN

```

Note

Always use forward slashes when referring to file paths, even on Windows.

The first statement in the **log4j.properties** file sets the logging level to INFO and defines three appenders: stdout, serverAccess, and serverError. Setting the logging level to INFO configures the logging mechanism such that it only logs events with a severity of INFO or higher. The logging severity in ascending order is: DEBUG, INFO, WARN, ERROR, and FATAL. To log all events, set the logging level to DEBUG.

Appender properties allow you to control the way that log information is formatted and filtered. The following table shows some of the important properties.

Property name	Description
CategoryExclude	Comma-separated list of logging categories. Only log events whose category isn't in this list are logged.
CategoryInclude	Comma-separated list of logging categories. Only log events with the specified categories are logged.
Delimiter	The delimiter character to use between field values. Valid values are

	tab , space , or the actual delimiter character.
EventExclude	Comma-separated list of logging categories. Only log events whose event name isn't in this list are logged.
EventInclude	Comma-separated list of logging events. Only log events with the specified event name are logged.
Field	Comma-delimited list of fields to log.
OutputHeader	Boolean value (true/false) that instructs the logging system to write out a W3C ECLF header whenever the server is started.
QuoteFields	Boolean value (true/false) that instructs the logging system to wrap field data in double quotes.

For more information about how to configure the log4j specific properties such as log file rolling and additional log appender types, see the [Log4j website](#).

Wowza Streaming Engine can also be configured to generate logs on a per-application and per-virtual host basis. These configurations are included, but commented-out, at the bottom of the **default [install-dir]/conf/log4j.properties** file. The first commented-out section includes configuration for per-application logging. The second commented-out section includes configuration for per-virtual host logging. To enable either of these features, remove the comments (**#** sign at the beginning of each of the lines) from the section.

The per-application logging generates log files using the following directory structure:

```
[install-dir]/logs/[vhost]/[application]/wowzastreamingengine_access.log
[install-dir]/logs/[vhost]/[application]/wowzastreamingengine_error.log
[install-dir]/logs/[vhost]/[application]/wowzastreamingengine_stats.log
```

The per-virtual host logging generates log files using the following directory structure:

```
[install-dir]/logs/[vhost]/wowzastreamingengine_access.log
[install-dir]/logs/[vhost]/wowzastreamingengine_error.log
[install-dir]/logs/[vhost]/wowzastreamingengine_stats.log
```

This method for generating log files can be very useful if you want to offer Wowza Streaming Engine as a shared service to several customers.

More resources

The Wowza website contains a trove of articles that offer step-by-step instructions for configuring common streaming scenarios and implementing all of the features and capabilities of Wowza Streaming Engine. Here's a selection of useful and popular articles, as well as a link to the Wowza Streaming Engine doc landing page.

- [Set up video-on-demand streaming in Wowza Streaming Engine](#)
- [Set up live streaming using an RTMP-based encoder in Wowza Streaming Engine](#)
- [Set up live streaming using an RTSP/RTP-based encoder in Wowza Streaming Engine](#)
- [Publish and play a live stream \(MPEG-TS based encoder\) in Wowza Streaming Engine](#)
- [Connect a live source to Wowza Streaming Engine](#)
- [Set up and run Transcoder in Wowza Streaming Engine](#)
- [Set up and run Wowza nDVR in Wowza Streaming Engine](#)
- [Configure a live stream repeater in Wowza Streaming Engine](#)
- [Re-stream video from an IP camera \(RTSP/RTP re-streaming\) in Wowza Streaming Engine](#)
- [Stream over MPEG-DASH with Wowza Streaming Engine](#)
- [All Wowza Streaming Engine articles](#)