



Wowza Streaming Engine™

User's Guide

Wowza Streaming Engine: User's Guide



Version: 4.5

<https://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 Streaming Cloud™, Wowza Streaming Engine™, along with other trademarks, logos, trade dress, and other proprietary colors and markings, are each 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.

Document History

Version	Description	Date
Doc v4.5.0	Document release for Wowza Streaming Engine 4.5.0	05-23-2016

Note

A more recent version of this document may be available online. See the [Wowza Media Systems User Guides webpage](#) for the latest updates.

Table of Contents

What's New	7
New Stream Target Destinations and Capabilities.....	7
New Integrated Live Sources from our Works with Wowza Partners	8
New Media Cache Source	8
WebM Support for MPEG-DASH Streams.....	9
WebSocket Server	9
SNMP Server.....	9
APIs for Monitoring and Extracting Live MPEG-TS Stream Data.....	9
LATM Encapsulation.....	10
Introduction	11
Adobe HDS (Adobe Flash Player)	11
Apple HLS (iPhone, iPad, iPod touch, QuickTime, and More)	12
Microsoft Smooth Streaming (Microsoft Silverlight and More)	13
MPEG-DASH Streaming (DASH Clients).....	14
Adobe RTMP (Adobe Flash Player).....	15
RTSP/RTP (QuickTime, VLC, 3GPP Devices, Set-top Boxes, and More)	16
Video and Audio Streaming, Recording, and Chat	17
Live Stream Transcoding and Transrating	17
Live Stream DVR Playback.....	20
Stream Encryption with DRM.....	20
AddOns.....	21
Installed Examples.....	23
Wowza Streaming Engine Editions	24
Server Installation	25
Before Installation	25
Installing Wowza Streaming Engine	26
Starting and Stopping the Software	27
Uninstalling Wowza Streaming Engine	31
Running Wowza Streaming Engine as a Named User.....	31
Running Multiple Wowza Streaming Engine Instances.....	32
Entering a New License Key.....	32
Ports Used for Streaming.....	35
Server Configuration and Tuning	36
Software Updates	37

Application Configuration.....	38
Applications and Application Instances (Application.xml)	39
URL Formats	39
Stream Types	40
HTTP Streamers and Live Stream Packetizers	42
Timed Text Providers.....	45
Transcoder and nDVR Configurations	47
Modules	47
Properties	49
Media Types	50
Content Storage.....	51
Advanced Configuration Topics	53
MediaCasters, Stream Files, and Startup Streams.....	53
Live Stream Repeater (Origin/Edge Live Streaming)	58
Live Stream Recording	60
Virtual Hosting	61
Server-side Publishing (Stream and Publisher Classes)	65
Server Management and Monitoring	66
Starting and Stopping Wowza Streaming Engine Manager	66
Managing Sign-In Credentials.....	70
Navigating in Wowza Streaming Engine Manager	71
Adobe Flash Streaming and Scripting.....	84
Streaming Basics	84
Pre-built Media Players.....	86
Bi-directional Remote Procedure Calls	87
Remote Shared Objects.....	88
Server Administration	90
Configuring SSL and RTMPS	90
Logging	91
Streaming Tutorials	98

What's New

What's new in the latest Wowza media server software?

Wowza Streaming Engine™ is robust, customizable, and scalable media server software that powers reliable streaming of high-quality audio and video to any device anywhere. This release of the server software has new and updated features that continue to future-proof and simplify online video delivery by expanding its ability to stream video to any screen using any streaming protocol, streamlining media server administration, and providing enhanced scalability, flexibility, and reliability.

New Stream Target Destinations and Capabilities

New destinations: Facebook, SHOUTcast, and Icecast

The **Stream Targets** feature in Wowza Streaming Engine enables you to send live streams to content delivery networks (CDNs), streaming servers, streaming services, and multicast networks for distributed delivery. The destination distributes your live stream, enabling you to scale your Streaming Engine implementation, computing resources, and delivery options for a variety of scenarios. Wowza Streaming Engine 4.5.0 has new destination options that enable you to send live video streams to **Facebook** and live audio-only streams to **SHOUTcast** and **Icecast** servers. For more information, see [How to stream to Facebook Live](#) and [How to stream to SHOUTcast and Icecast](#).

Akamai: Apple HLS ingest and egress over HTTPS

On the Akamai HD network, you can configure endpoints that ingest and deliver all content over HTTPS. Wowza Streaming Engine 4.5.0 has new properties for Apple HLS destination options that take advantage of this capability for Apple HLS streams. You can configure a property to send Apple HLS streams to the Akamai HD network over HTTPS. You can also

configure a property to specify that the Apple HLS playlist and chunks are delivered to players over HTTPS. For more information, see [How to stream to the Akamai HD network](#).

Apple HLS delivery to generic stream target

Wowza Streaming Engine 4.5.0 also introduces the ability to create a generic stream target that receives an Apple HLS stream from Wowza Streaming Engine. This feature is aimed at developers who want to do the following:

- Send HLS streams to destinations using any desired path and/or communications protocol.
- Modify and customize the playlist and chunks using the Java API.

For more information, see [How to send Apple HLS streams to a generic destination](#).

New Integrated Live Sources from our Works with Wowza Partners

Wowza Streaming Engine software has integrated publishing options that simplify live streaming workflows. The **Sources (Live)** feature for live applications in Wowza Streaming Engine Manager enables you to deliver connection settings for the application to a variety of popular encoders and cameras.

Wowza Streaming Engine 4.5.0 adds **Panasonic** camcorders and PTZ cameras to its growing lineup of integrated encoders and cameras. For more information, see the following articles:

- [How to connect Panasonic AJ Series camcorders to Wowza Streaming Engine](#)
- [How to connect Panasonic AW Series PTZ cameras to Wowza Streaming Engine](#)

To see our entire lineup of encoders and cameras that are integrated with Wowza Streaming Engine software, see our [Works with Wowza webpage](#).

New Media Cache Source

The **Media Cache** feature in Wowza Streaming Engine is a read-through caching mechanism that increases the scalability of on-demand video streaming for your Wowza media server configuration. In this release of Wowza Streaming Engine software, we've increased the flexibility of Media Cache deployment options by adding **Google** Cloud Storage Buckets as a supported media cache source for media files. You can configure Wowza Streaming Engine Manager to request media files from a single bucket linked to a Google Cloud service

account. For more information, see [How to scale video on demand streaming with Media Cache](#).

WebM Support for MPEG-DASH Streams

Wowza Streaming Engine 4.5.0 supports the WebM file format and codecs. WebM streams are compressed with the VP8 and VP9 video codecs and the open-source Vorbis and Opus audio codecs. Your live applications in Wowza Streaming Engine can ingest WebM-encoded source streams or you can transcode an H.264-encoded source stream to the WebM format. For more information, see [How to transcode live streams using WebM](#).

Note

The WebM file format and codecs were supported as a technology preview feature in previous versions of the Wowza Streaming Engine software. We recommend that you update the software to Wowza Streaming Engine 4.5.0 for the best results.

WebSocket Server

The WebSocket is a protocol that provides full-duplex communication changes or a single TCP connection. WebSocket is supported by most modern web browsers and can be used with socket servers such as node.js. You can use a new HTTP Provider in Wowza Streaming Engine 4.5.0 to host WebSocket sessions. For more information, see [How to create a WebSocket server](#).

SNMP Server

Wowza Streaming Engine 4.5.0 adds support for Simple Network Management Protocol version 2c (SNMPv2c), a network management protocol that you can use to collect information about your Wowza Streaming Engine media server configuration. This lets you add your Wowza media servers to your SNMP-managed network and monitor status for different media server SNMP objects from your SNMP management system. For more information, see [How to use SNMP in Wowza Streaming Engine](#).

APIs for Monitoring and Extracting Live MPEG-TS

Stream Data

MPEG-TS live streams can carry multiple streams of data including audio, video, closed captions, ad markers (SCTE-35, SCTE-104), key length value (KLV), and more. Each stream is identified by a packet identifier (PID). Wowza Streaming Engine 4.5.0 includes an API for connecting a custom listener to the MPEG-TS ingestion process to insert PID monitors into the stream and a PID

monitor API for extracting the stream data whenever data is available on a given PID. The code examples in [How to monitor MPEG-TS ingestion to process additional data streams](#) show how to add an MPEG-TS ingestion listener API and PID monitor API to your media server to extract additional stream data and implement a built-in monitor for SCTE-35 markers and KLV data. Listening to other types of MPEG-TS data is also possible using these APIs.

LATM Encapsulation

In Wowza Streaming Engine 4.5.0, we added support for Low Overhead Audio Transport Multiplex (LATM) encapsulation of AAC over MPEG-2 transport streams.

Introduction

What is Wowza Streaming Engine?

Wowza Streaming Engine™ is high-performance, extensible, and fully interactive media streaming software platform that provides live and on-demand streaming, chat, and remote recording capabilities to a wide variety of media player technologies. The Wowza Streaming Engine software can deliver content to many popular media players such as Adobe Flash Player; Microsoft Silverlight player; Apple iPhone, iPad, and iPod touch and Apple QuickTime player (version 10 or later); Android smartphones and tablets; and IPTV/OTT set-top boxes. Wowza Streaming Engine software includes support for many streaming protocols including Adobe HTTP Dynamic Streaming (Adobe HDS), Apple HTTP Live Streaming (Apple HLS), Microsoft Smooth Streaming, MPEG-DASH streaming, MPEG-2 Transport Streams (MPEG-TS), Real Time Messaging Protocol (RTMP), Real Time Streaming Protocol (RTSP), and Real-time Transport Protocol (RTP). It's an alternative to the Adobe Media Server, Darwin Streaming Server, Microsoft IIS Media Services, and other media servers.

For the most up-to-date information, tutorials, and tips, see the [Wowza Articles and Forums](#).

To get up and running, see the [Wowza Streaming Engine Quick Start Guides](#).

Adobe HDS (Adobe Flash Player)

Wowza Streaming Engine software can stream adaptive bitrate live and video on demand (VOD) content to Adobe Flash Player 10.1 or later using the Adobe HTTP Dynamic Streaming (Adobe HDS) protocol. 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 the Wowza Streaming Engine server. Adobe 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. The Wowza Streaming Engine software supports the following video and audio codecs when using this streaming 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)

Apple HLS (iPhone, iPad, iPod touch, QuickTime, and More)

Wowza Streaming Engine software can stream adaptive bitrate live and VOD H.264, AAC, and MP3 content to iOS-based devices (iPhone/iPad/iPod touch iOS version 3.0 or later), QuickTime player (version 10 or later), Safari browser (version 4.0 or later), and other devices such as the Roku and Amino set-top boxes and some brands of smart TVs using the Apple HTTP Live Streaming (Apple HLS) protocol. Apple HLS 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 the Wowza Streaming Engine server. Apple HLS is referred to as "Cupertino" streaming in the Streaming Engine configuration files.

Wowza Streaming Engine software supports multiple encryption methods for protecting Apple HLS streams using DRM. See [How to secure Apple HLS streaming using DRM encryption](#).

Wowza Streaming Engine software can send timed data events to the iOS player in the form of ID3 tags. See [How to convert OnTextData events in a live or vod stream to timed events \(ID3 tags\) in an Apple HTTP stream](#).

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

The iPhone, iPad, and iPod touch (iOS devices) and Apple TV digital media extender support the following media formats:

Video

- H.264

Audio

- AAC, AAC Low Complexity (AAC LC), High Efficiency AAC (HE-AAC) v1
- Dolby Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3)
- MP3

Microsoft Smooth Streaming (Microsoft Silverlight and More)

Wowza Streaming Engine software can stream adaptive bitrate live and VOD H.264, AAC, and MP3 content to Microsoft Silverlight, Windows Phone devices, and other devices using the Microsoft Smooth Streaming protocol. Microsoft Silverlight is a cross-browser, cross-platform technology that exists on many personal computing devices. 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 the Wowza Streaming Engine server 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

MPEG-DASH Streaming (DASH Clients)

Dynamic Adaptive Streaming over HTTP (DASH), also known as *MPEG-DASH*, is a new international standard for adaptive streaming that's being adopted by the streaming industry. Wowza Streaming Engine software includes MPEG-DASH technology for streaming live and VOD content over HTTP to select DASH clients.

MPEG-DASH is similar to proprietary adaptive streaming technologies such as Apple HLS, Adobe HDS, and Microsoft Smooth Streaming in that it's a chunk-based streaming technology that uses HTTP for delivery. All media-chunking and packaging necessary to deliver a stream using this technology is performed by the Wowza Streaming Engine server. Note that in MPEG-DASH terminology, chunks are called *segments*.

MPEG-DASH servers give DASH clients a list of the available media chunk URLs in a Media Presentation Description (MPD) manifest. The MPD describes chunk information such as timing, language, timed text, and media characteristics (video resolution and bitrate). Clients request media chunks sequentially based on network conditions, device capabilities, and other factors to enable uninterrupted playback of the adaptive bitrate media presentation.

The MPEG-DASH standard ([ISO/IEC 23009-1](#)) defines segment container formats for ISO Base Media File Format (ISOBMFF) and MPEG-2 Transport Streams (MPEG-2 TS). MPEG-DASH is codec-agnostic and supports multiplexed and non-multiplexed encoding. Multiple content protection (DRM) schemes are supported; however, a Common Encryption (CENC) standard ([ISO/IEC 23001-7](#)) is being developed in conjunction with MPEG-DASH to allow content to be encrypted once and then streamed to DASH clients that support different licensing systems.

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

Video

- H.264
- VP8
- VP9

Audio

- AAC, AAC Low Complexity (AAC LC), AAC High Efficiency (HE-AAC) v1 and v2
- Dolby Digital 5.1 Surround Sound (AC-3) and Dolby Digital Plus (Enhanced AC-3 or E-AC-3)
- Vorbis
- Opus

For more information about MPEG-DASH support in Wowza Streaming Engine software, see [How to do MPEG-DASH streaming](#).

Adobe RTMP (Adobe Flash Player)

Wowza Streaming Engine software communicates with Adobe Flash Player using the Real Time Messaging Protocol (RTMP). The Wowza Streaming Engine software can deliver adaptive bitrate live and VOD content to Flash Player using RTMP and it supports other features such as shared objects, video recording, video chat, remote procedure calls, and more. The Wowza Streaming Engine software 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 software 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 the Wowza Streaming Engine server 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 Monthly Edition and Perpetual Edition licenses for Wowza Streaming Engine software can use [Wowza StreamLock™ AddOn](#) to get free 256-bit SSL certificates for use with RTMP (RTMPS) and HTTP (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 software includes bi-directional support for Action Message Format (AMF3 and AMF0) for data serialization (AMF3 was introduced in Flash Player 9 and ActionScript 3.0).

RTSP/RTP (QuickTime, VLC, 3GPP Devices, Set-top Boxes, and More)

Wowza Streaming Engine software 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. The Wowza Streaming Engine software can also accept 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, the Wowza Streaming Engine software 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 software 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 software 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 about how to use the query parameters, see [How to specify per-stream settings in .stream files](#).

Video and Audio Streaming, Recording, and Chat

Wowza Streaming Engine software can stream live and VOD content to many player technologies. It supports the following VOD file formats: MP4 (QuickTime container - .mp4, .f4v, .mov, .m4a, .m4v, .mp4a, .mp4v, .3gp, and .3g2), FLV (Flash Video - .flv), and MP3 content (.mp3). The Wowza Streaming Engine software can accept live video and audio streams from sources that support the RTMP, RTSP/RTP, native RTP, and MPEG-TS protocols and it can record any live source stream to either the MP4 or FLV format.

Wowza Streaming Engine software 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 software 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 software can deliver two-way video, audio, and text chat to Adobe Flash Player. This feature can be leveraged to deliver video conferencing applications or two-way messaging applications.

Live Stream Transcoding and Transrating

The Transcoder feature in Wowza Streaming Engine software is a real-time video transcoding and transrating solution that provides the ability to ingest a live stream, decode the video and audio, and then re-encode the stream for delivery to desired playback devices. It can decode and re-encode audio and video in multiple formats with key frames 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

The 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
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	

* = Preview Transcoder Technology

Transcoder support for the H.265 codec is provided as an early feature preview of software that may be updated in a later release of Wowza Streaming Engine. Unexpected results can occur when using feature preview software. To get the best results when using this codec, see the following resources on our website:

- To use the **H.265** video codec, follow the instructions in [How to stream using HEVC/H.265 transcoding](#).
- You can't use Wowza Streaming Engine Manager to specify that the H.265 codec be used for your transcoded output renditions. Instead, you must edit your Transcoder template files in a text editor. For details, see the [XML reference](#) in "How to set up and run Wowza Transcoder for live streaming."

Hardware acceleration

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

The Transcoder can be configured to take advantage of 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 Transcoder](#).
- **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 the Transcoder, and links to configuration instructions, see [Server specifications for NVIDIA NVENC and NVIDIA CUDA acceleration with Wowza Transcoder](#).

Note

This release of the Wowza Streaming Engine software has support for encoding H.265/HEVC video renditions using the NVIDIA **NVENC** hardware acceleration encoding option. This is a technology preview feature that may be updated in a later release of the software.

Overlays

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 stationary image effects such as a watermark to your video. In addition, you can use a Java-based API to apply dynamic overlay images to streams. The API can be configured manually or pre-programmed based on external events, making it a powerful tool for adding premium TV-like experiences. See [How to add graphic overlays to live streams with Wowza Transcoder](#).

For more information about the Transcoder feature, see the [Wowza Transcoder User's Guide](#) and the [Wowza Transcoder Forum](#).

Live Stream DVR Playback

The nDVR feature in Wowza Streaming Engine software provides the ability to record a live stream into a cache on the media server. This enables viewers that join the live stream in-progress to 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 the Wowza Streaming Engine server.

For more information about the nDVR feature, see the [Wowza nDVR User's Guide](#) and the [Wowza nDVR Forum](#).

Stream Encryption with DRM

The DRM feature in Wowza Streaming Engine software provides integration 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 Apple HTTP Live Streaming (HLS).

Integration is supported for the following Key Management Service providers:

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

For more information about the DRM feature, see the [Wowza DRM online tutorials](#) and the [Wowza DRM Forum](#).

Note

Wowza Streaming Engine software has APIs that enable encryption schemes for on-the-fly encryption of live and VOD Apple HLS streams, including **SAMPLE-AES** (sample-level encryption for version 5 of the Apple 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). The Wowza Streaming Engine software also has an API that enables on-the-fly encryption of live and VOD Microsoft Smooth Streaming format with PlayReady protection for INSIDE Secure player technology. The DRM feature in Wowza Streaming Engine software isn't required to use these APIs. For more information, see:

- [How to secure Apple HLS streaming using DRM encryption](#)
- [How to protect streams for delivery to INSIDE Secure player technology](#)

AddOns

Wowza™ provides the following AddOn packages that you can download and install to extend and enhance Wowza Streaming Engine server functionality.

AddOn Package	Description
Load Balancing	The Dynamic Load Balancing AddOn enables you to distribute HTTP, RTMP, and RTSP streams across multiple Wowza edge servers dynamically. The edge servers communicate with one or more load-balancing Wowza servers and clients connect to the load-balancing server to get the least-loaded edge server. See How to get the Dynamic Load Balancing AddOn .
Central Configuration	The Central Configuration AddOn provides a system for managing multiple Wowza media servers in a complex environment from a central location. You can modify the system to fit most CDN and service provider environments. See How to get Central Configuration AddOn (simplify multiple server deployments) .

Network Security	The Wowza StreamLock™ AddOn is 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 software . StreamLock-provisioned SSL certificates provide the best security when used with RTMP. The certificates can also be used for secure HTTP streaming (HTTPS). The certificates expire after one year. See How to get SSL certificates from the StreamLock service .
GeoIP Locking	The GeoIP AddOn enables you to restrict access to streamed content based on a client's geographic location. See How to enable geographic locking .
Stream Name Aliasing	The StreamNameAlias AddOn 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 How to get the StreamNameAlias AddOn .
Load Testing	You can use the Flash RTMP Load Test Tool to generate RTMP load on a single Wowza Streaming Engine instance to test configuration and performance. The Load Test Tool requires a subscription or perpetual license for the Wowza Streaming Engine software. See How to get Flash RTMP Load Test Tool .
Module Collection	The Wowza Module Collection is a collection of ready-to-use modules and utilities that provide specific functionality for Wowza-based applications. The modules in the collection are already compiled, so Java experience isn't required. However, source code is also provided with all of the modules to allow easy modification for specific use cases. See Module Collection .

Note

For an up-to-date list of the AddOn packages and information about how to use them, see the [AddOns webpage](#).

Installed Examples

The Wowza Streaming Engine software includes the following examples that highlight the server functionality. The examples are located in **[install-dir]/examples**. The **[install-dir]/examples/README.html** file describes the available examples and how to install them.

Example	Description
VideoOnDemandStreaming	This example shows how to configure and playback video on demand (VOD) content. It includes sample players for Apple iOS devices, Adobe Flash, Microsoft Silverlight, and DASH clients and source code for an OSMF-based Flash Player and Microsoft Silverlight 3 or later. It uses the default stream type.
LiveVideoStreaming	This example shows how to configure and playback live video. It includes sample players for iOS devices, Adobe Flash, Microsoft Silverlight, and DASH clients and source code for an OSMF-based Flash Player and Microsoft Silverlight 3 or later. It uses the live stream type.
LiveDVRStreaming	This example shows how to configure the nDVR feature in a Wowza Streaming Engine server to record and playback a live video with DVR. It includes sample players for Adobe Flash and Microsoft Silverlight and source code for Microsoft Silverlight 3 or later. It uses the live stream type.
SHOUTcast	This Adobe Flash example shows how to re-stream SHOUTcast MP3 or AAC+ audio data through a Wowza Streaming Engine server. It uses the shoutcast stream type.
VideoChat	This Adobe Flash example shows how to implement video chat between two users. It uses the live-lowlatency stream type for RTMP and the Camera and Microphone objects to get video and audio content. The example can stream video and audio data between two client connections or loop the data back to itself.

WebcamRecording	This Adobe Flash example shows how to implement the advanced client-to-server video-recording capabilities in a Wowza Streaming Engine server using Adobe Flash Player. It uses the record stream type and the Camera and Microphone objects to get video and audio content. To use this example, you'll need a web camera (webcam) and Adobe Flash running in a web browser.
ServerSideModules	Developers can use this example with the Wowza Integrated Development Environment (Wowza IDE) to learn how to create custom server-side modules. The example contains server-side module class files and Flash client applications that demonstrate how a Wowza Streaming Engine server interacts with Flash clients.

Notes

- All Adobe Flash examples are implemented using ActionScript 3.0.
- Stream types are used to control the different types of streaming (live, VOD, chat, remote recording, and so on.) See [Stream Types](#).

Wowza Streaming Engine Editions

Subscription (Monthly Edition) and lifetime (Perpetual Edition) licenses are available for use with Wowza Streaming Engine software to accommodate nearly any use case or business need. See [Wowza Streaming Engine Pricing](#).

Server Installation

How do I install Wowza Streaming Engine?

Wowza Streaming Engine™ is a small but powerful Java server. This chapter describes how to install and run the Wowza Streaming Engine software. Wowza Streaming Engine software automatically installs a server version of the Java Runtime Environment (JRE) that it requires, making it easy to get a Wowza Streaming Engine server up and running.

Notes

- If you can't or don't want to use the version of Java that installs with Wowza Streaming Engine software, you can use a different JRE. Wowza Streaming Engine software requires a JRE that supports deploying Java in a server environment. Java 6, 7, and 8 are all supported but Java 8 is recommended. For more information on how to install a different, supported version of Java and how to instruct a Wowza Streaming Engine server to use it, see [How to manually install and troubleshoot Java on your Wowza media server](#).
- The [Transcoder feature](#) in Wowza Streaming Engine software is available only for 64-bit Windows and Linux operating systems running a 64-bit Java VM.

Before Installation

If you're upgrading your Wowza Media Server™ software (version 3.6 or earlier) to Wowza Streaming Engine, you must uninstall Wowza Media Server before installing Wowza Streaming Engine. For instructions, see [How to upgrade Wowza Media Server software to Wowza Streaming Engine](#).

If you're updating your earlier version of Wowza Streaming Engine software to Wowza Streaming Engine, you shouldn't run the installer. Instead, run the updater. For instructions, see [How to update your Wowza Streaming Engine installation](#).

Installing Wowza Streaming Engine

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

1. 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 installer file, **WowzaStreamingEngine-4.5.0-windows-installer.exe**, and follow the wizard instructions.
 - **OS X** – Double-click the disc image file, **WowzaStreamingEngine-4.5.0-osx-installer.dmg**, and then double-click the installer package icon, **WowzaStreamingEngine-4.5.0-osx-installer**, and follow the wizard instructions.
 - **Linux** – In the folder where you downloaded the Linux package, run the commands below for your platform. Then, follow the wizard instructions.

64-bit Linux Systems

```
sudo chmod +x WowzaStreamingEngine-4.5.0-linux-x64-  
installer.run  
sudo ./WowzaStreamingEngine-4.5.0-linux-x64-installer.run
```

32-bit Linux Systems

```
sudo chmod +x WowzaStreamingEngine-4.5.0-linux-installer.run  
sudo ./WowzaStreamingEngine-4.5.0-linux-installer.run
```

During the installation process, you'll be asked:

- To accept the terms of the license agreement.
- To 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 the Wowza Streaming Engine software installed, you may be able to use the license key value found in the **[install-dir]/conf/Server.license** file.
- To create a user name and password for an Administrator account. You'll use this account to sign in to the browser-based Wowza Streaming Engine Manager. The user name and password values are case-sensitive.

- To confirm or change the install location. By default, Wowza Streaming Engine software installs in the following directories:
 - **Windows** – /Program Files (x86)/Wowza Media Systems/Wowza Streaming Engine 4.5.0/
 - **OS X** –
 - /Applications/Wowza Streaming Engine 4.5.0/
 - /Library/LaunchDaemons/
 - /Library/WowzaStreamingEngine/ (an alias)
 - /Library/WowzaStreamingEngine-4.5.0/
 - **Linux** – /usr/local/WowzaStreamingEngine-4.5.0/ (as the **root** user)
- If you want to start the Wowza Streaming Engine server and Wowza Streaming Engine Manager automatically after the installation finishes. Accept the default option **Start Wowza Streaming Engine automatically**. This option configures the server and manager software to start automatically as system services. If you don't choose this option, you must start the server software and the manager manually before you can use the software. See [Starting and Stopping the Software](#) and [Starting and Stopping Wowza Streaming Engine Manager](#).

Note

A silent installation option is available for all platforms, including Red Hat Package Manager and Debian Package Manager options for Linux. Please note that approval is required to activate this option. For more information, contact sales@wowza.com.

Starting and Stopping the Software

Wowza Streaming Engine software 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 server software and Wowza Streaming Engine Manager install as system services, which means you're running an active instance of Wowza Streaming Engine software 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 the 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 software 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.

Note

Wowza Streaming Engine software 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 4.5.0** 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 4.5.0** and then click **Stop**.

Wowza Streaming Engine software 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 4.5.0**, 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 the Wowza Streaming Engine software 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

- The **Wowza Streaming Engine 4.5.0** service runs under the **Local System account** by default. This can limit how the Wowza Streaming Engine software 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 the Transcoder feature is only available when running Wowza Streaming Engine as a Windows standalone application. It's not available when the Wowza Streaming Engine software is invoked as a service.

Start and stop Wowza Streaming Engine as a service (OS X)

To start the service, double-click the **Start Services** application in **/Applications/WowzaStreamingEngine-4.5.0** 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-4.5.0** 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 (OS X)

To start the software, double-click the **Start Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.5.0** or open a terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.5.0/bin
./startup.sh
```

To stop the software, double-click the **Stop Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.5.0** or open a terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.5.0/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 [How to manually install and troubleshoot Java on your Wowza media server](#).

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 4.5.0**, and then click **Uninstall**.
- **OS X** – Go to **/Applications/Wowza Streaming Engine 4.5.0** and double-click **Wowza Streaming Engine Uninstall**.
- **Linux** – In the folder where you downloaded the Linux package, run the following command:

```
sudo usr/local/WowzaStreamingEngine/uninstall
```

Running Wowza Streaming Engine as a Named User

On OS X and Linux platforms, the default installation of Wowza Streaming Engine software runs the server as the **root** user. If you want to run the server as a different user, follow the instructions in [How to run Wowza Streaming Engine as a Named User \(Linux and OS X\)](#) 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 the Wowza Streaming Engine server on a lower-numbered ports such as 80 (HTTP), 443 (HTTPS, RTMPS), or 554 (RTSP), the server must continue to run as the **root** user.

Running Multiple Wowza Streaming Engine Instances

You can run multiple instances of the same version of Wowza Streaming Engine software on one computer instead of running one large instance with multiple virtual hosts (VHosts). An *instance* is defined as a single copy of the running Streaming Engine software. This is useful in cases where additional resources are available on the computer but can't be used by a single instance, your streaming workflow requires a custom design that won't work using a single instance, or you're testing a multiple-server deployment on a single computer. Each instance requires a valid license. You can use a single Monthly Edition license key for multiple instances and each instance is counted separately and reflected in your bill. See [How to run multiple instances of Wowza Streaming Engine on one computer](#).

Entering a New License Key

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

```
Windows - %WMSCONFIG_HOME%\conf\Server.license
OS X - /Library/WowzaStreamingEngine/conf/Server.license
Linux/Unix - /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 the Wowza Streaming Engine software. If you need to enter multiple license keys (for example, for the media

server software and for the Transcoder, nDVR, and DRM technologies), 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 is restarted.

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.

Pro Edition Licenses for Wowza Streaming Engine

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

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

Server Setup

Basic Properties Server Listeners

[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
EPBP4-XXXXX-XXXXX-XXXXX-XXXXX-h5t3C

Default Stream Prefix
mp4

Options


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

RTP Datagram Starting Port
6970

For licensing details, see [Wowza Streaming Engine Perpetual Pro Edition pricing information](#).

The Monthly Subscription Pro Edition license is best for variable demand. You can install as many instances of the Streaming Engine software as needed using the same license key and enable the Transcoder, nDVR, and DRM technologies integrated with each instance. A Monthly Subscription Pro Edition 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-jy6vu

Default Stream Prefix
mp4

Options

☒ Enable monitoring and statistics for this server

☐ Allow RTP datagram port sharing

RTP Datagram Starting Port
6970

For Monthly Subscription Pro Edition pricing information, see [Pay-as-you-go Monthly Billing](#).

Notes

- If you purchased a Perpetual Pro Edition license for Wowza media server software before December 23, 2015, your license key has an **EPBU4** prefix and it licenses the Streaming Engine software and the Wowza Transcoder and Wowza 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 the media server software).
- If you purchased a license for Wowza media server software before January 1, 2015, contact sales@wowza.com to learn more about how to license the software.

Trial Edition Licenses for Wowza Streaming Engine

A [Trial Edition license](#) for the Wowza Streaming Engine software expires after 180 days. When you acquire a new, paid license for the software, you must delete the Trial Edition 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 Edition 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 the media server software or re-create its settings when you replace the Trial Edition license key with a paid license for the software.


Ports Used for Streaming

Before streaming with Wowza Streaming Engine software, you should open ports on your firewall. The following table shows the default ports that the Wowza Streaming Engine server uses for streaming. All of these port numbers are configurable through the configuration files that are described later in this document.

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

By default, a Wowza Streaming Engine server 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

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 software can't share ports with other programs or services, so make sure that there are no other programs or services running that share the added ports.

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

TCP 80	Adobe HDS, Apple HLS, Microsoft Smooth Streaming, MPEG-DASH 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 the Wowza Streaming Engine configuration settings by editing the XML files directly, be sure to review the [Wowza Streaming Engine Configuration Reference](#), 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 software 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, the server 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 [Performance Tuning](#).

Software Updates

In between production releases, development builds are produced periodically in the form of updates. This allows users to get early access to new features in the latest Wowza Streaming Engine software releases and to give feedback. Information about what's included in each update is included in a README.txt file that's included in the update archive (.zip) file. For more information about how to apply an update to your server software, see [How to update your Wowza Streaming Engine installation](#).

Application Configuration

How do I create and configure an application for streaming?

Wowza Streaming Engine™ software is designed to handle multiple streaming protocols. All streaming is controlled through the creation and configuration of streaming applications. One application can be configured to simultaneously deliver either live or video on demand (VOD) content to multiple player technologies. 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**.
5. The **myapplication** page is displayed so that you can configure the application settings.

A single application can be configured to deliver a live or video on demand (VOD) stream at the same time to Adobe Flash Player, Apple iOS devices (iPhone, iPad, or iPod touch) or Apple TV digital media extender, Roku and Amino set-top boxes, Microsoft Silverlight, DASH clients, and RTSP/RTP-based players (including 3GPP smartphones and tablets, and Android devices). The [Tutorials](#) section of the Wowza™ website has step-by-step instructions for configuring applications for common streaming scenarios. The remainder of this chapter covers application configuration details. For more detailed configuration information, see the [Wowza Streaming Engine Configuration Reference](#).

Applications and Application Instances

(Application.xml)

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 and 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 is loaded, it looks in the following locations for an **Application.xml** file (where **[application]** is the application name):

```
[install-dir]/conf/[application]/Application.xml
[install-dir]/conf/Application.xml
```

The first **Application.xml** file that's found is used.

URL Formats

All streaming in the Wowza Streaming Engine server 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 Adobe Flash Player, Apple iOS devices, Microsoft Silverlight, DASH clients, or RTSP/RTP, follow a similar format:

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

-where-

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

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

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

The following are example URLs for different player technologies. The examples assume that a live video with the stream name **myStream** using the application name **live** is streamed.

Adobe HDS

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

Apple HLS

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

Microsoft Smooth Streaming

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

MPEG-DASH Streaming

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

Adobe RTMP

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

RTSP/RTP

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

Now is probably a good time to take a quick look at the default settings for applications. The rest of this chapter describes the most commonly configured items.

Stream Types

Wowza Streaming Engine software uses named stream types to control the different types of streaming (live, VOD, chat, remote recording, and so on.). Stream types are automatically configured when you create different application types and configure their options in Wowza Streaming Engine Manager. You can also edit the **Streams/StreamType** property in **Application.xml** using a text editor to change the stream type for an application. The following table shows the stream types and their uses.

Stream type	Description
default	VOD
file	VOD
live	Publish and play live content (best for one-to-many streaming of live events)
live-lowlatency	Publish and play live content over RTMP (best for one-to-one or one-to-few video/audio chat applications)
live-record	Same as live —in addition content is recorded
live-record-lowlatency	Same as live-lowlatency —in addition content is recorded
liverepeater-edge	Publish and play live content across multiple Wowza servers in an origin/edge configuration (used to configure edge application)
liverepeater-edge-lowlatency	Publish and play live content across multiple Wowza servers in an origin/edge configuration (used to configure edge application when latency is important)
liverepeater-edge-origin	Publish and play live content across multiple Wowza servers in an origin/edge/edge configuration (used 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 latency is important
rtp-live-record	Same as rtp-live —in addition content is recorded
rtp-live-record-lowlatency	Same as rtp-live-lowlatency —in addition content is recorded
shoutcast	Re-stream SHOUTcast/Icecast MP3 or AAC+ audio streams
shoutcast-record	Same as shoutcast —in addition content is 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 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
Adobe HDS	Enables the application to stream live and VOD content to Flash Player using the Adobe HTTP Dynamic Streaming (HDS) protocol. It adds the sanjosestreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
Apple HLS	Enables the application to stream live and VOD content to iOS-based devices (iPhone/iPad/iPod touch iOS version 3.0 or later), QuickTime player (version 10 or later), Safari browser (version 4.0 or later), and to other devices such as Roku and Amino set-top boxes and some brands of smart TVs, using the Apple HTTP Live Streaming (HLS) protocol. It adds the cupertinostreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
Microsoft Smooth Streaming	Enables the application to stream live and VOD content to Microsoft Silverlight using the Microsoft Smooth Streaming protocol. It adds the smoothstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
MPEG-DASH	Enables the application to stream live and VOD content to DASH clients using the Dynamic Adaptive Streaming over HTTP (DASH) protocol. It adds the mpegdashstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .

nDVR (live streaming only)	When you enable the nDVR feature for live or live http origin applications in Wowza Streaming Engine Manager, it enables the application to stream live content from Wowza Streaming Engine (origin) to Wowza Streaming Engine (edge). It adds the dvrchunkstreaming HTTP streamer to the <HTTPStreamers> section in Application.xml .
-----------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Live streams coming into a Wowza Streaming Engine server must be packaged (*packetized*) before they can be delivered to clients using 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 server to clients. They're also used when delivering a live stream from an origin Wowza server to an edge Wowza 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
Adobe HDS	Enables Adobe HDS live streaming to Flash Player. It adds the sanjosestreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
Apple HLS	Enables Apple HLS live streaming to iOS-based devices. It adds the cupertinostreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
Microsoft Smooth Streaming	Enables Microsoft Smooth Streaming to Silverlight. It adds the smoothstreaming streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .
MPEG-DASH	Enables MPEG-DASH streaming to DASH clients. It adds the mpegdashstreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml .

nDVR	When you enable the nDVR feature for live or live http origin applications in Wowza Streaming Engine Manager, it adds the dvrstreamingpacketizer streaming packetizer to the <LiveStreamPacketizers> section in Application.xml for use with nDVR.
-------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Repeater packetizers are used when delivering a live stream from a Wowza 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
Adobe HDS	Enables Adobe HDS live stream repeater for Flash Player. It adds the sanjosestreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
Apple HLS	Enables Apple HLS live stream repeater for iOS-based devices. It adds the cupertinostreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
Microsoft Smooth Streaming	Enables Microsoft Smooth Streaming live stream repeater for Silverlight. It adds the smoothstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
MPEG-DASH	Enables MPEG-DASH live stream repeater for DASH clients. It adds the mpegdashstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml .
nDVR	When you enable the nDVR feature for live edge applications in Wowza Streaming Engine Manager, it adds the dvrstreamingrepeater repeater packetizer to the <LiveStreamPacketizers> section in Application.xml for use with nDVR.

For more information about how to implement the live stream repeater (origin/edge) mechanism for delivering a live media event across multiple Wowza servers, see [Live Stream Repeater \(Origin/Edge Live Streaming\)](#).

Note

The nDVR feature in Wowza Streaming Engine software 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. This capability can be extended to an edge Wowza server in an origin/edge configuration. See the [Wowza nDVR User's Guide](#).

Timed Text Providers

Wowza Streaming Engine software includes support for timed text (closed captioning) for live and video on-demand streams. The Wowza Streaming Engine server enables you to convert caption data from a variety of instream and file-based sources to appropriate caption formats for live and on-demand video streaming using the Adobe HDS, Apple 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 software 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 Apple HLS streams or as **onTextData** events in Adobe HDS and Adobe RTMP streams. In addition, instream CEA-608 caption data can be passed through the Transcoder for delivery in Apple 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	This option 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 Apple HLS streams. It adds the ModuleOnTextDataToCEA608 module to the <Modules> section in Application.xml .

Embedded CEA-608 captions in live streams	This option enables the application to monitor live streams for CEA-608 captions, decode the captions, and convert them to onTextData events in Adobe HDS and Adobe RTMP streams. It 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. This option enables the application to monitor live streams for AMF onTextData events and pass them through in Adobe HDS and Adobe RTMP streams. It adds the captionLiveIngest property to the <TimedText>/<Properties> section in Application.xml .

See [How to configure closed captioning for live streaming](#).

Closed Captioning for Video on Demand Streams

For video on-demand streams, Wowza Streaming Engine software 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). The ingested captions can be delivered as CEA-608-formatted SEI data in Apple HLS streams or as Action Message Format (AMF) **onTextData** events in Adobe HDS and Adobe 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	This option enables the application to pull captions directly from 3GPP tracks (codecID "tx3g") that are embedded in MP4 VOD assets. This option is enabled by default. It adds the vodcaptionprovidermp4_3gpp timed text provider to the <TimedText> section in Application.xml .

Timed Text (TTML/DXFP) file	This option 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. It adds the vodcaptionproviderttml timed text provider to the <TimedText> section in Application.xml .
SubRip (SRT) file	This option 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. It adds the vodcaptionprovidersrt timed text provider to the <TimedText> section in Application.xml .
Web Video Text Track (WebVTT) file	This option 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. It adds the vodcaptionproviderwebvtt timed text provider to the <TimedText> section in Application.xml .
Scenarist Closed Caption (SCC) file	This option 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. It adds the vodcaptionproviderscc timed text provider to the <TimedText> section in Application.xml .

See [How to configure closed captioning for video on demand streaming](#).

Transcoder and nDVR Configurations

The **<Transcoder>** and **<DVR>** containers in an **Application.xml** file serve to configure an application to use the Transcoder and nDVR features in Wowza Streaming Engine software. See the [Wowza Streaming Engine Configuration Reference](#) and the following tutorials:

- [How to set up and run Wowza Transcoder for live streaming](#)
- [How to set-up and run Wowza nDVR for live streaming](#)

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 [Server-side modules](#).

In the 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** information is for providing a detailed description of the module and isn't used in any operations. The **Class** item 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. The Wowza Streaming Engine software comes with many modules that can be added to the **Modules** list to provide additional functionality. For a complete list of the modules, see [Built-in server-side modules](#). You can also use the [Wowza Integrated Development Environment \(Wowza IDE\)](#) to develop your own custom modules to provide additional functionality. See [How to extend Wowza Streaming Engine using Java](#).

Notes

- Access to the **Modules** tab is limited to administrators with advanced permissions. See [Managing Sign-In Credentials](#).
- Wowza provides a collection of utility modules that are ready to use in Wowza applications. These modules don't require you to use the Wowza IDE. See [Module Collection](#).

Properties

Properties are a list of 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. In the **Application.xml** configuration file, a property definition has the following form:

```
<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, you can 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](#).

Many articles on the Wowza website prescribe custom properties for tuning the server and for adding advanced functionality. When adding custom properties, it's important to add them to the correct **<Properties>** container in **Application.xml**. The 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 software 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 appending 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.

The Transcoder feature in Wowza Streaming Engine software uses a templating system to group 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 **nggrp:** media type prefix.

Wowza Streaming Engine software has an API that can be used to intercept requests for adaptive bitrate streams and provide the stream grouping through Java API calls. 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 the Wowza Streaming Engine server reads a SMIL file, it creates a MediaList and passes it back to the streaming provider. This API provides a means for intercepting the requests and creating the MediaList dynamically in a Wowza Streaming Engine module. See [How to use Java API calls to resolve SMIL file requests \(AMLST\)](#).

Content Storage

By default Wowza Streaming Engine software is configured to stream VOD content from (and record 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, you can 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) files 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, you can 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:

<code>\${com.wowza.wms.AppHome}</code>	- Application home directory
<code>\${com.wowza.wms.ConfigHome}</code>	- Configuration home directory
<code>\${com.wowza.wms.context.VHost}</code>	- Virtual host name
<code>\${com.wowza.wms.context.VHostConfigHome}</code>	- Virtual host configuration directory
<code>\${com.wowza.wms.context.Application}</code>	- Application name
<code>\${com.wowza.wms.context.ApplicationInstance}</code>	- Application instance name

Advanced Configuration Topics

How do I take advantage of Wowza Streaming Engine features?

This chapter covers more advanced streaming topics. Some of the functionality discussed is provided by [AddOn packages](#). AddOn packages are downloadable packages that include server extensions for adding more advanced features to the Wowza Streaming Engine™ software.

MediaCasters, Stream Files, and Startup Streams

Wowza Streaming Engine software includes a system for re-streaming live streams called *MediaCaster*. The MediaCaster system is used to re-stream IP camera streams (RTSP/RTP streams), SHOUTcast/Icecast streams, and streaming output from native RTP or MPEG-TS encoders. The MediaCaster system pulls a stream from a stream source and makes it available for streaming to all player technologies supported by the Wowza Streaming Engine software. This 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, the MediaCaster system 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 methodology works great for RTMP and RTSP/RTP streaming where advanced packetization isn't required. However, the model doesn't work for the HTTP streaming protocols (Adobe HDS, Apple HLS, Microsoft Smooth Streaming, and MPEG-DASH streaming). An Apple iOS device requires 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.

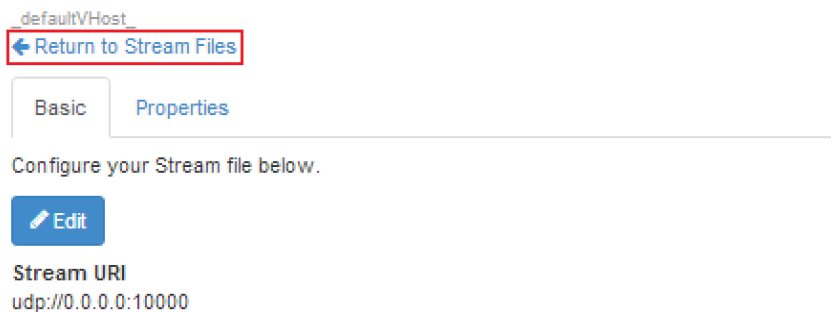
Stream Files

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





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

X

Stream Name

mycoolevent.stream

Application Name

live

Application Instance

☒ Connect to default application instance: _default_

☐ 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

Be sure to select the **MediaCaster Type** in the list that corresponds to the source stream type:

- Select **rtp** for IP Camera streams (RTSP/RTP streams) and for streams from native RTP and MPEG-TS encoders.
- Select **shoutcast** for SHOUTcast/Icecast streams.
- Select **liverepeater** if the stream is pulled from another server that's running Wowza Media Server or Wowza Streaming Engine software.

You can then use **mycoolevent.stream** in the following example URLs to play the stream:

Adobe HDS

```
http://[wowza-ip-address]/live/mycoolevent.stream/manifest.f4m
```

Apple HLS

```
http://[wowza-ip-address]/live/mycoolevent.stream/playlist.m3u8
```

Microsoft Smooth Streaming

```
http://[wowza-ip-address]/live/mycoolevent.stream/Manifest
```

MPEG-DASH

```
http://[wowza-ip-address]/live/mycoolevent.stream/manifest.mpd
```

Adobe RTMP

```
Server: rtmp://[wowza-ip-address]/live  
Stream: mycoolevent.stream
```

RTSP/RTP

```
rtsp://[wowza-ip-address]/live/mycoolevent.stream
```

Note

In the **SMIL Files** feature 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 the **Startup Streams** feature in Wowza Streaming Engine Manager to create stream entries in the **[install-dir]/conf/StartupStreams.xml** file. Stream entries in this file are automatically started when the server is started (or more specifically, when a virtual host is started). 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]:
  - name of live application that re-streams the source stream

[mediacaster-type]:
  - valid mediacaster type: rtp, rtp-record, shoutcast, shoutcast-record,
  liverepeater

[stream-name]:
  - name of the source stream
```

As an 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**.

Add to Startup Streams
X

Stream Name

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

See [How to start streams at server startup](#).

Note

The following server-side methods can also be used to start and stop streams using the MediaCaster system:

```
IApplicationInstance.startMediaCasterStream (...);
IApplicationInstance.stopMediaCasterStream (...);
```

For more information about these methods, see the [Wowza Streaming Engine Server-Side API](#).

Live Stream Repeater (Origin/Edge Live Streaming)

A live stream repeater uses multiple Wowza servers in an origin/edge configuration to deliver live media content across multiple servers. The encoded media content is delivered to the origin server in the same manner as if you were delivering the content to a single Wowza server. A player requests the content 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.

The example in this section 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 3 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:

Adobe HDS

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

Apple HLS

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

Microsoft Smooth Streaming

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

MPEG-DASH

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

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 the **Stream Files** feature 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

- The WOWZ™ protocol is a TCP-based messaging protocol in Wowza Streaming Engine software and is used for server-to-server communication. It's enabled by default. If one of the Wowza servers in the origin/edge configuration is running a version of the software that doesn't support the WOWZ protocol, an RTMP connection is established between that server and other servers instead.
- You can secure the connection between Wowza servers in and origin/edge configuration by using a SecureToken shared secret. See [How to configure a live stream repeater](#).
- 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 the **Startup Streams** feature 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 [Dynamic Load Balancing](#).

Live Stream Recording

There are multiple ways to record live streams to VOD files for later playback, but the **Incoming Streams** feature 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 [How to record live streams](#).

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, you must 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. The **Stream Files** and **Startup Streams** features 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 the Wowza Streaming Engine software. 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 also give you 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 the **Incoming Streams** feature 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

The Wowza Streaming Engine software 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, a Wowza server is configured with a single VHost named **_defaultVHost_**.

Configuration Files

The **VHosts.xml** configuration file in the Wowza Streaming Engine **[install-dir]/conf** folder is used to define each of the VHost environments. 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. See [Typical Configuration](#) to view a sample directory structure.
- **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

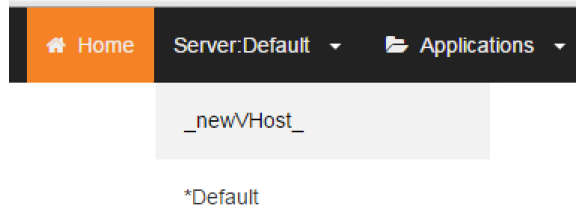
- By default, all VHost environments share the **publish.password** file for the default VHost. You can use the **Source Authentication** feature in Wowza Streaming Engine Manager to set up unique publishing credentials for each VHost and the unique credentials are stored in this file.

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 the **Source Authentication** feature in the manager to update the file. See [Custom password file location](#).

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

VHosts are defined in the `[install-dir]/conf/VHosts.xml` file. 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 software 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.



It's important to note that Wowza Streaming Engine software 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.

Server-side Publishing (Stream and Publisher Classes)

Wowza Streaming Engine software 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 [How to schedule streaming with Wowza Streaming Engine \(ServerListenerStreamPublisher\)](#).

The **Publisher** class is a low-level publishing API that lets you inject raw compressed video and audio frames into the Wowza Streaming Engine server to create a custom live stream. See the **Publisher** class server API Javadocs (`[install-dir]/documentation/serverapi`) for the current detailed documentation. The article [How to use Publisher API and JSpeex to publish an audio stream \(VOIP integration\)](#) includes an audio sample that walks through the process of publishing Speex data to a stream.

Server Management and Monitoring

How do I manage and monitor Wowza Streaming Engine?

Wowza Streaming Engine™ Manager enables you to conveniently set up, manage, monitor, and measure video and audio streams using a web browser on your computer, tablet, or phone. The new browser-based application extends the programmatic and command line configuration and management of the Wowza Streaming Engine software, 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. On Windows operating systems, if you have multiple browsers installed on your computer, you can ensure that the web application always opens in the browser that you want to use by configuring the **Default Programs** feature.

Starting and Stopping Wowza Streaming Engine Manager

Notes

- Wowza Streaming Engine software 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, you can open it in a web browser with the following URL: `http://[wowza-ip-address]:8088/enginemanager`. Where **[wowza-ip-address]** is the Wowza Streaming Engine IP address or domain name.
- On Windows operating systems, you can open Wowza Streaming Engine Manager in the default browser from the **Start** menu (**Start > All Programs > Wowza Streaming Engine 4.5.0 > Wowza Streaming Engine Manager**).
- 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 4.5.0**, 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 4.5.0**, 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 4.5.0**, 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 the 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 (OS X)

To start the service, double-click the **Start Services** application in **/Applications/WowzaStreamingEngine-4.5.0** 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-4.5.0** 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 (OS X)

To start the manager in standalone mode, double-click the **Start Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.5.0** or open a terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.5.0/manager/bin
./startmgr.sh
```

To stop the manager, double-click the **Stop Standalone Mode** application in **/Applications/WowzaStreamingEngine-4.5.0** or open a terminal window and enter the following commands:

```
cd /Library/WowzaStreamingEngine-4.5.0/manager/bin
./shutdownmgr.sh
```

Note

The **Start Standalone Mode** and **Stop Standalone Mode** applications also start and stop Wowza Streaming Engine software 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 the 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 the 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 the Wowza Streaming Engine software. This account has administrator access to enable control of the Wowza Streaming Engine server through the manager. However, it doesn't provide access to advanced property, server listener, and module settings, which are reserved for expert Wowza 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.

To 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**. As the signed-in user, you'll be signed-out automatically 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.

To create new 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 in Wowza Streaming Engine Manager

This section introduces the different parts of the Wowza Streaming Engine Manager browser-based application to help you find your way around the user interface. For additional details, see [How to 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. A yellow banner at the top right contains a performance warning. The main content area is divided into several sections: a 'Welcome' message, a 'Monthly Subscription' status, a 'Status' section with charts for connections and usage, a 'Server Uptime' section, a 'Test Video' section, an 'Application Connection Settings' section, and a 'Getting Started With Applications' section. Numbered callouts (1-5) point to specific elements: 1 points to the navigation tabs, 2 points to the Status section, 3 points to the Application Connection Settings, 4 points to the Test Video section, 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.

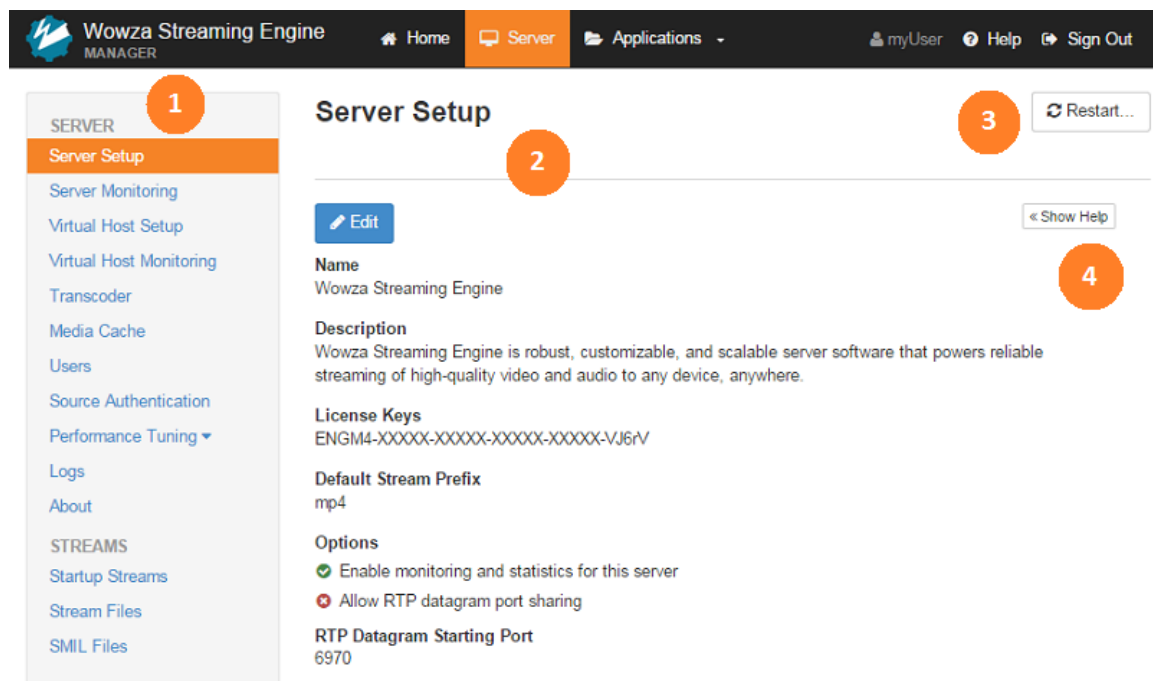
2 View information in the **Status** area about how the total number of connections (both source and playback connections) for the server (the **Connections** chart) and the total server resource consumption for CPU, Java heap, memory, and disk (the **Usage** chart). You can also see if the Transcoder, nDVR, and DRM features are licensed, and if they're enabled, 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 over multiple streaming protocols.

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



- 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 for the Wowza Streaming Engine instance such as the instance name, available license keys, and enabling/disabling the Monitoring features 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 [How to monitor server connections, load, and application statistics](#).

Virtual Host Setup: Manage virtual hosting environments on the server. By default, the Wowza Streaming Engine software ships with a single VHost environment named `_defaultVHost_`; however, you can add more VHost environments and manage them separately with this feature. See [Virtual Hosting](#).

Virtual Host Monitoring: Monitor VHost source and playback connections, network throughput, and uptime. See [How to monitor server connections, load, and application statistics](#).

Transcoder: Monitor the number of concurrent live source streams ingested by the Transcoder and add, modify, and delete the Transcoder templates. See the [Wowza Transcoder User's Guide](#).

Media Cache: Configure the read-through caching mechanism that enables scaling of video on demand (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 [How to scale video on demand streaming with Media Cache](#).

Users: Set up and manage administrator and read-only user accounts for Wowza Streaming Engine Manager. See [Managing Sign-In Credentials](#).

Source Authentication: Create and manage case-sensitive 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 server performance settings from the default values that are calculated when the server starts. See [How to do performance tuning](#).

Logs: View the messages in Wowza Streaming Engine and Wowza Streaming Engine Manager log files directly in the manager. Filtering and display options let you customize what's displayed and the UI provides a way for you to download large log files to a compressed (zipped) folder for viewing outside of the manager. See [How to view log messages in Wowza Streaming Engine Manager](#).

About: View information about the Wowza Streaming Engine platform such as the installed Wowza Streaming Engine software version and license and the installed Java Runtime Environment (JRE).

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 server 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](#).

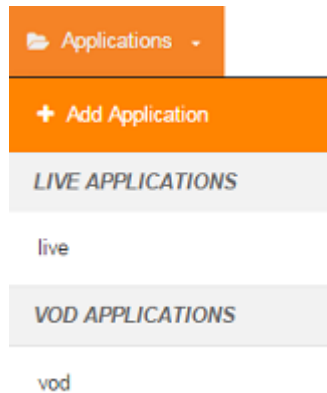
SMIL Files: Create Synchronized Multimedia Integration Language (SMIL) files that organize streams of various bitrates into groups for HTTP adaptive bitrate streaming. See [How to do adaptive bitrate streaming](#).

- 2 When you click a server feature in the contents panel, a page is displayed that enables you to configure the feature settings. Advanced settings for fine-tuning the server configuration are available for some of the 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. You can click the **Hide Help** button to hide this information. If the information is hidden, you can click the **Show Help** button to bring it back into view.

Application Types

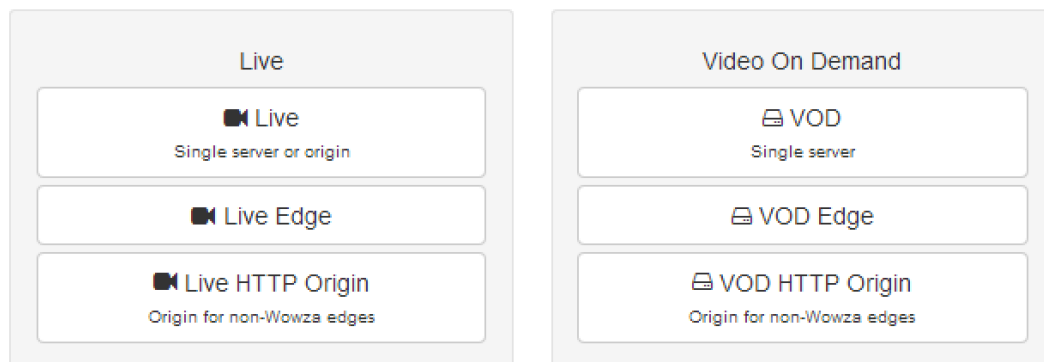
An *application* is a set of configuration options in a Wowza Streaming Engine server that supports a specific use case for the delivery of streaming content. To add applications in Wowza Streaming Engine Manager, click the **Applications** tab and then click **Add Application**.



In the **Add Application** page that's displayed, you can add applications for six streaming use cases.

Add Application

Select an Application Type.



Live

Use this application to deliver live streams to players (single server) or as an origin server to deliver live streams to other servers running Wowza Media Server™ or Wowza Streaming Engine software to scale content delivery to a large number of players.

VOD

Use this application to stream video on demand (VOD) files to players (single server).

Live Edge

Use this application to ingest live streams from a live application on an origin server that's running Wowza Media Server or Wowza Streaming Engine software. This application is then used to deliver the live streams to players (single server).

VOD Edge

Use this application to ingest video on demand files from a Media Cache source. This application is then used to stream the VOD files to players (single server).

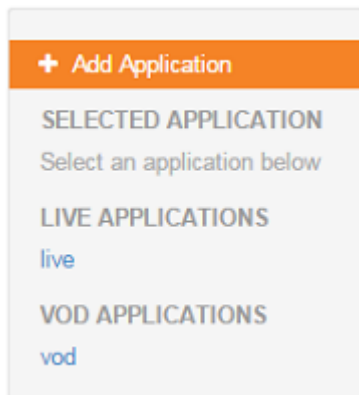
Live HTTP Origin

Use this application to deliver live streams to an HTTP caching infrastructure using HTTP streaming protocols (MPEG-DASH, Apple HLS, Adobe HDS, and Microsoft Smooth Streaming).

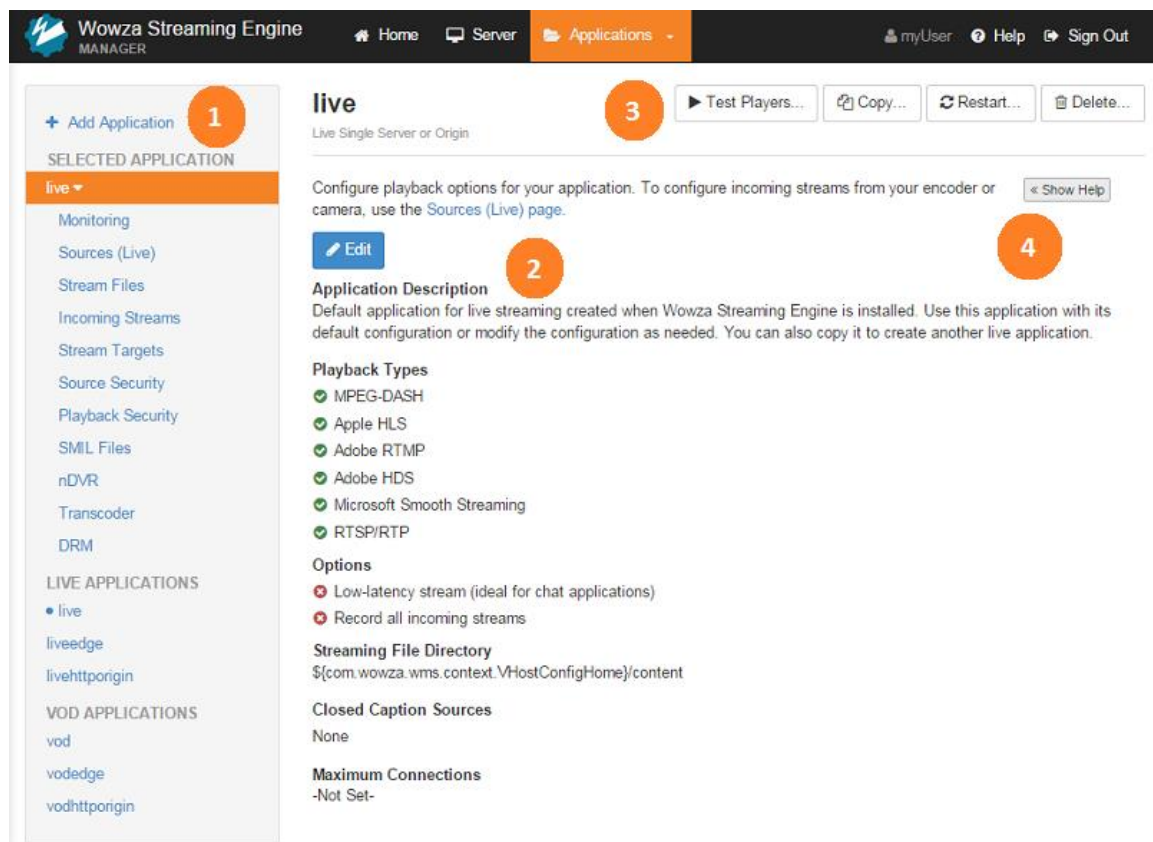
VOD HTTP Origin

Use this application to deliver video on demand files to an HTTP caching infrastructure using HTTP streaming protocols (MPEG-DASH, Apple HLS, Adobe HDS, and Microsoft Smooth Streaming).

To add an application, click the **Application Type** in the page that corresponds to your use case, 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 software.



Application Configuration



- 1 The contents panel provides access to the following features that let you configure, manage, and monitor the different 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 [How to monitor server connections, load, and application statistics](#). 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 software supports integrated configuration of All live application types

additional live sources provided by [Works With Wowza partners](#). You can connect many other live sources to live Wowza Streaming Engine applications, but you must configure the connection settings manually. See [How to 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 video on demand (VOD) files for later playback. See [How to record live streams](#).

All live application types

Stream Targets: Send live streams to content delivery networks (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 a Wowza Streaming Engine server. 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 [How to do adaptive bitrate streaming](#).

Live
Live Edge
VOD
VOD Edge

nDVR: Configure DVR playback of live streams using the nDVR feature. See the [Wowza nDVR User's Guide](#).

All live application types

Transcoder: Configure transcoding of live streams to suit desired playback devices using the Transcoder feature. See the [Wowza Transcoder User's Guide](#).

Live
Live HTTP Origin

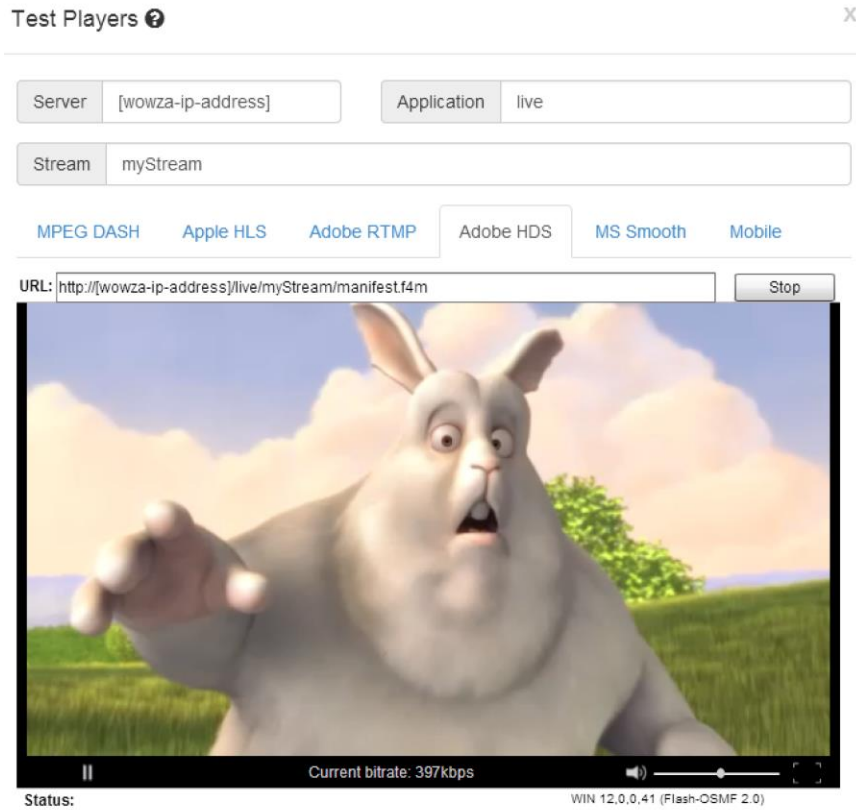
DRM: Integrate with DRM Key Management Service	Live
partners to enable on-the-fly DRM encryption of premium	Live Edge
live and VOD content for a variety of playback devices. See	VOD
Stream Encryption with DRM.	VOD Edge

- 2 When you click an application or one of its features in the contents panel, a page is displayed 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. You can click the **Hide Help** button to hide this information. If the information is hidden, you can click the **Show Help** button to bring it back into view.

Test Players

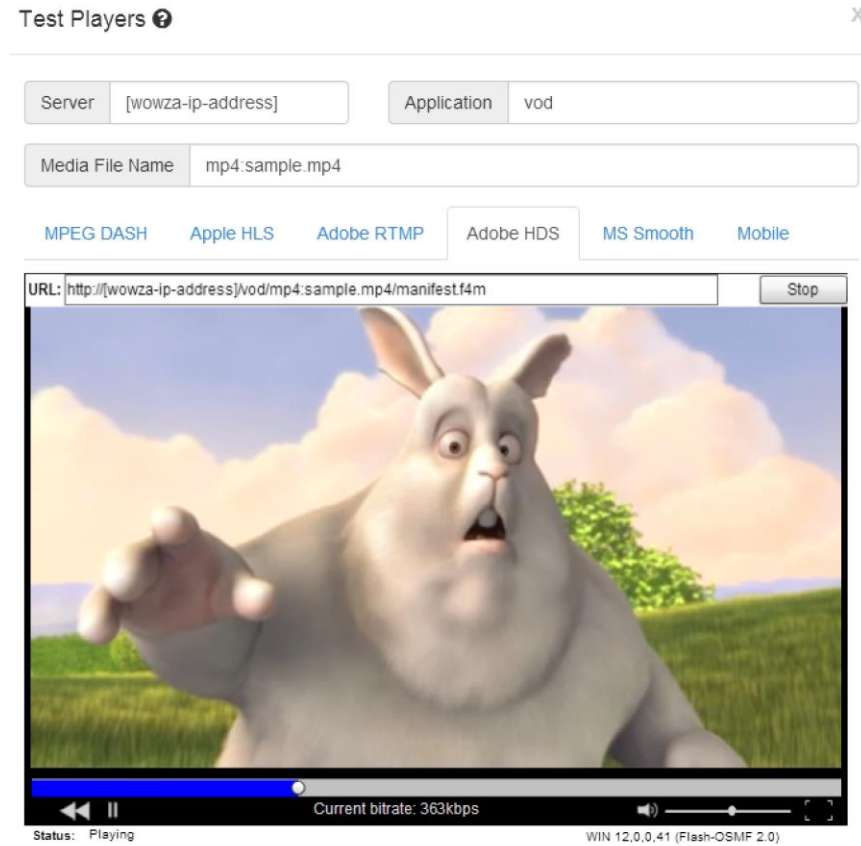
Wowza Streaming Engine Manager provides test players for all of the live and on demand application types so that you can test an application's streaming configuration. To access the test players, click the **Test Players** button in the upper-right corner of the application or feature page. Then in the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

The test players for live applications are preconfigured to play a live stream named **myStream** from 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 on demand applications are preconfigured to play the **[install-dir]/content/sample.mp4** video file that's installed with the server software.



If you want to use your own VOD file, you can copy it to the **[install-dir]/content** root folder and substitute its file name in place of **sample.mp4** in the **Media File Name** box. 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]/myVideos**, enter **vod/_definst_/myVideos** in the **Application** box.

Note

The test players can't display closed captions or playback encrypted streams. DVR playback is only supported by the Adobe HDS, Apple 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 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 only view (and not 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. You can click a link in the **Quick Links** area at the top of the page to jump to the associated property settings. For example, you can 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
Cupertino Streaming Packetizer	MPEG-DASH Streaming Packetizer	Closed Captions
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 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](#).

Adobe Flash Streaming and Scripting

What can I do with Wowza Streaming Engine and Adobe Flash Player?

The Wowza Streaming Engine™ software includes additional features that are only applicable to Adobe Flash Player when using the RTMP protocol (or any of its variants).

When used with Adobe Flash Player, the Wowza Streaming Engine software is more than just a streaming server—it's an application server. It provides features such as shared objects, video chat, remote recording, and bi-directional remote procedure calls.

Streaming Basics

We'll start with the most basic code that's needed to play a live or video on demand stream in Flash. Assume that we've followed the instructions in [How to set up video on demand streaming](#) and that we have an application named **vod** that's configured to stream on-demand video. In Adobe Flash Creative Suite 3 or later, do the following:

1. Create a new **Flash File** with ActionScript 3.0 support.
2. Open the library palette (On the **Window** menu, select **Library**).
3. Right-click in the library palette, and then select **New Video**. Enter **video** in **Symbol name**, and then click **OK** to create the video object.
4. Drag the **video** object from the library to the stage, and then in the properties palette, give it an instance name of **video1**.
5. In **Window > Actions**, select **Scene 1**.
6. Enter the following code:

```

var nc:NetConnection = new NetConnection();
var ns:NetStream = null;

function ncOnStatus(infoObject:NetStatusEvent)
{
    trace("ncOnStatus: "+infoObject.info.code);
    if (infoObject.info.code == "NetConnection.Connect.Success")
    {
        trace("Connection established");
        ns = new NetStream(nc);

        ns.bufferTime = 3;

        video1.attachNetStream(ns);

        ns.play("mp4:sample.mp4");
    }
}
nc.addEventListener(NetStatusEvent.NET_STATUS, ncOnStatus);

nc.connect("rtmp://localhost/vod");

```

7. On the **Debug** menu, select **Debug Movie**.

You should now be streaming the **sample.mp4** video file. This is the most basic ActionScript 3.0 code that's needed for live and video on demand playback. If you inspect the code, you'll see how simple it is. We create a **NetConnection** object for streaming and add an event listener so that we can be notified when the connection to Wowza Streaming Engine server is established. When we're notified of a successful connection, we create a **NetStream** object and start to playback the stream.

The **LiveVideoStreaming** and **VideoOnDemandStreaming** example players that are installed with the Wowza Streaming Engine software take this example a little further. The example players support progress bars, pause, stop, and full screen. Inspecting the code for the example players is a good next step for learning how to stream. The **VideoChat** and **WebcamRecording** examples are a great starting point to learn how to publish video and audio using the built-in **Camera** and **Microphone** objects. See the [Installed Examples](#) section in this document.

Pre-built Media Players

Building your own custom player with advanced functionality can be a daunting task. Another option is to use pre-built Flash video players. This section describes a few of the more popular Adobe Flash Player options.

Adobe FLVPlayback component

The Adobe FLVPlayback component is a pre-built video player component that you can add to your own Flash project. It includes features such as play, pause, seek, stop, and full screen. It comes with Adobe Flash CS3 or later. The component is updated occasionally, so it's best to keep your Adobe Flash software up-to-date to ensure that you're running the most recent version. The nice thing about this component is that it can be integrated into your custom Flash code.

JW Player

[JW Player](#) is a pre-built Flash-based player. It includes a rich set of features such as playlists, skinning, closed-captioning, and ad-serving. It's fully supported and there's a commercial option. It also includes a built-in version of the Wowza SecureToken security mechanism.

For more information about how to use JW Player with Wowza Streaming Engine software, see the following support articles:

- [How to use JW Player with Wowza Streaming Engine](#)
- [How to add SecureToken protection to JW Player](#)

Flowplayer

[Flowplayer](#) is an open source pre-built Flash-based player. It includes a rich set of features similar to JW Player. It also includes a built-in version of the Wowza SecureToken security mechanism.

For more information about how to use Flowplayer with Wowza Streaming Engine software, see [How to use Flowplayer with Wowza Streaming Engine](#).

Strobe Media Playback player

The Strobe Media Playback player supports RTMP streaming and Adobe HDS streaming. The player is built on the Open Source Media Framework (OSMF) and is hosted by Adobe. See [How to use Strobe Media Playback with a Wowza media server](#).

Bi-directional Remote Procedure Calls

Wowza Streaming Engine software supports bi-directional remote procedure calls to and from Adobe Flash Player. Bi-directional remote procedures calls are a way for ActionScript code running in Flash Player to invoke server-side Java methods and pass data to a Wowza Streaming Engine server. The server can, in turn, invoke client-side ActionScript methods and pass data to Flash Player. This enables you to build rich client/server applications using Flash Player and Wowza Streaming Engine software. These features are available when using the RTMP protocol.

Calls from Flash Player to a Wowza Streaming Engine server are performed using the following method:

```
NetConnection.call(methodName, resultObj, params...)
```

For example, the following ActionScript 3.0 client-side code calls the server-side method **doSomething**, passes the parameters **value1** and **value2**, and receives a single return value:

```
function onMethodResult(returnVal:String):Void
{
    trace("onMethodResult: "+returnVal);
}
nc.call("doSomething", new Responder(onMethodResult), value1, value2);
```

Note

See [Custom module classes](#) for the server-side code for this method.

Receiving method calls from a Wowza Streaming Engine server is done by adding handler methods/functions to the client object that's attached to the **NetConnection** object. For example, the following ActionScript 3.0 client-side code adds the handler method **onSomethingHappened** that receives two string parameters **value1** and **value2**:

```
var clientObj:Object = new Object();
clientObj.onSomethingHappened(value1:String, value2:String):Void
{
    trace("onSomethingHappened: "+value1+"-"+value2);
}
nc.client = clientObj;
```

For more information about the programming model, see [How to extend Wowza Streaming Engine using Java](#).

Remote Shared Objects

Wowza Streaming Engine software supports Adobe Flash remote shared objects (RSOs), which enable data-sharing between the Wowza Streaming Engine server and multiple Flash Players. Remote shared objects are an extension of ActionScript objects that enable shared object data to be synchronized between Adobe Flash Players on the same or different client machines. Shared data is synchronized by the Wowza Streaming Engine server through an event-based synchronization method. RSOs can also be persisted on the server to maintain data across sessions.

Each Flash Player that subscribes to a shared object is notified when the shared object data is updated. Shared object data can be changed client-side by Flash Player or server-side through the Wowza Streaming Engine **ISharedObject** API. The following example shows the ActionScript 3.0 code that's needed to create a remote shared object and set a value:

```
var nc:NetConnection = new NetConnection();
var test_so:SharedObject = null;
var timer:Timer = null;
function ncOnStatus(infoObject:NetStatusEvent)
{
    trace("ncOnStatus: "+infoObject.info.code);
    if (infoObject.info.code == "NetConnection.Connect.Success")
    {
        test_so = SharedObject.getRemote("test", nc.uri);
        test_so.addEventListener(SyncEvent.SYNC, syncEventHandler);
        test_so.connect(nc);

        timer = new Timer(1000, 1);
        timer.addEventListener(TimerEvent.TIMER, setSOProperty);
        timer.start();
    }
}
function syncEventHandler(ev:SyncEvent)
{
    trace("syncEventHandler");
    var infoObj:Object = ev.changeList;
    for (var i = 0; i < infoObj.length; i++)
    {
        var info:Object = infoObj[i];
        if (info.name != undefined)
            trace("  "+info.name+"="+test_so.data[info.name]);
        else
            trace("  [action]="+info.code);
    }
}
function setSOProperty(ev:TimerEvent):void
```

```
{  
    test_so.setProperty("testName", "testValue");  
}  
nc.addEventListener(NetStatusEvent.NET_STATUS, ncOnStatus);  
nc.connect("rtmp://localhost/vod");
```

Wowza provides a downloadable Adobe Flash example ([RemoteSharedObjects.zip](#)) that illustrates the basics of remote shared objects. It implements the basic remote shared object interface and the **onSync** event handler to highlight how data is synchronized between client connections.

Server Administration

How do I configure, manage, and deploy Wowza Streaming Engine?

Wowza Streaming Engine™ is a powerful Java server. It can be run standalone from a command shell or installed as a system service. Running the server standalone is best for developing custom Wowza Streaming Engine applications because the server can be started and stopped quickly and server log messages can be viewed immediately in the console window. Running the server 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 software supports Secure Sockets Layer (SSL) and RTMPS (RTMP over SSL) and HTTPS (HTTP over SSL) streaming protection. SSL is a technology that 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

- If you want to get an SSL certificate from Wowza for use with Wowza Streaming Engine software, see [How to get SSL certificates from the StreamLock service](#).
- If you want to get an SSL certificate from a certificate authority, see [How to request an SSL certificate from a certificate authority](#).
- If you want to create a self-signed SSL certificate, see [How to create a self-signed SSL certificate](#).

Logging

Wowza Streaming Engine software 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, the Wowza Streaming Engine server 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 more information about log messages, scenarios that may cause these messages, and suggestions for resolution, see [How to troubleshoot error messages](#).

Logging Fields

Wowza Streaming Engine software 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 (Apple HLS), http (Smooth Streaming), rtmp, rtmpe, rtmpe (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 software 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 netConnection.call("setStreamType", null, "[streamtype]")
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 software 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 for a Wowza Streaming Engine server 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.layout QuoteFields=false
log4j.appender.stdout.layout.Delimiter=space

# Access appender
log4j.appender.serverAccess=org.apache.log4j.DailyRollingFileAppender
log4j.appender.serverAccess.DatePattern='.'yyyy-MM-dd
log4j.appender.serverAccess.File=${com.wowza.wms.ConfigHome}/logs/wowzastreamingengine_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.DailyRollingFileAppender
log4j.appender.serverError.DatePattern='.'yyyy-MM-dd
log4j.appender.serverError.File=${com.wowza.wms.ConfigHome}/logs/wowzastreamingengine_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 the Windows platform).

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 software 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 software as a shared service to several customers.

Streaming Tutorials

Where do I get step-by-step instructions?

The [Tutorials](#) section of the Wowza website has step-by-step instructions for configuring common streaming scenarios. These instructions cover how to configure streaming to common player technologies such as Adobe Flash Player, Microsoft Silverlight, Apple iOS devices, and mobile devices. The following table briefly describes and provides links to online tutorials for common streaming scenarios.

Tutorial name	Description
How to set up video on demand streaming	Describes how to configure an application to stream video on demand (VOD) content.
How to set up live streaming using an RTMP-based encoder	Describes how to publish a live stream from RTMP-based encoders to a Wowza Streaming Engine server and how to configure an application to deliver the live stream.
How to set up live streaming using an RTSP/RTP-based encoder	Describes how to publish a live stream from RTSP/RTP-based encoders to a Wowza Streaming Engine server and how to configure an application to deliver the live stream.
How to set up live streaming using a native RTP encoder with SDP file	Describes how to use a live encoder that publishes a stream using Real-time Transport Protocol (native RTP) with Session Description Protocol (SDP) files to stream live content.
How to publish and play a live stream (MPEG-TS based encoder)	Describes how to use a live encoder that publishes a stream using the MPEG-2 Transport Stream (MPEG-2 TS) protocol to stream live content.

How to connect a live source to Wowza Streaming Engine	Describes how to publish live streams from encoders and cameras to a Wowza Streaming Engine server.
How to set up and run Wowza Transcoder for live streaming	Describes how to configure the Transcoder feature for a live application to transcode and transrate live source streams to multiple playback renditions that can be used in adaptive bitrate (ABR) streams.
How to set up and run Wowza nDVR for live streaming	Describes how to configure the nDVR feature for a live application to enable support for DVR playback of live source streams.
How to configure a live stream repeater	Describes how to configure live stream repeater (origin/edge) applications. Live stream repeater is a method for delivering a single live stream across a multiple server deployment to many viewers.
How to re-stream video from an IP camera	Describes how to re-stream and play a live stream from an IP camera.
How to re-stream audio from SHOUTcast/Icecast	Describes how to re-stream and play live SHOUTcast or Icecast audio streams.
How to set up live video recording	Describes how to configure an application for video recording using Flash Player.
How to set up live video chat	Describes how to configure an application for video chat using Flash Player.
How to do MPEG-DASH streaming	Describes how to configure video on-demand and live applications in a Wowza Streaming Engine server to deliver streams to Dynamic Adaptive Streaming over HTTP (DASH) clients.