Wowza Streaming Engine™

Configuration Reference

Wowza Streaming Engine: Configuration Reference

Version: 4.8

www.wowza.com

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

No Endorsement or Warranty for Third-Party Links and Software

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

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

Wowza Trademarks

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

Third-Party Trademarks and Copyrights

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

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

Document History

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

Note

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

Table of Contents

Chapter 1

Introduction

This reference guide describes the most commonly used settings and configuration items in the XML configuration files that are located in the **[install-dir]/conf** folder of the Wowza Streaming Engine™ media server software installation. Basic configuration settings are described, but the details of internal settings are omitted.

Chapter 2

Server Configuration Files

Server.xml

The **Server.xml** file configuration file is used to define and set server options that are independent of streaming applications that run on the media server.

Name

Name of the media server.

Description

Text description of the media server.

RESTInterface

Wowza Streaming Engine provides a REST Application Programming Interface (API) that you can use to configure and manage the server through HTTP requests. The **RESTInterface** settings are used to control access to the REST API. For more information, see Access reference documentation for the Wowza Streaming Engine REST API.

RESTInterface/[Enable, IPAddress, Port, AuthenticationMethod, DiagnosticURLEnable]

Enable: Boolean value that turns the REST interface for Wowza Streaming Engine on or off.

IPAddress: Network interface IP address that the REST interface binds to. If **IPAddress** is set to the wildcard (*****) character, the REST interface binds to all available network interfaces.

Port: Port used to access the REST interface (**8087**).

AuthenticationMethod: HTTP authentication method used to access the REST interface. This can be one of the following types: **none**, **basic**, **digest**, **digestfile**, or **remotehttp**. The default value is **digest**.

Example URLs for accessing the REST API:

- http://localhost:8087
- http://*username*:*password*@localhost:8087

DiagnosticURLEnable: Boolean value. When set to **true** (default), the following REST API URLs are available to users for diagnostic purposes:

- http://localhost:8087/
- http://localhost:8087/diag

Setting this value to **false** removes these URLs from the REST API.

RESTInterface/SSLConfig/[*]

Enable: Boolean value that specifies if access to the REST API should be made over a secure HTTP (HTTPS) connection using a Secure Sockets Layer (SSL) certificate. The default value is **false** (disabled).

KeyStorePath: Full path to the keystore file.

KeyStorePassword: Keystore password.

KeyStoreType: Keystore type. The default value is **JKS** for Sun Java JRE.

SSLProtocol: Cryptographic protocol. The default value is **TLS** (Transport Layer Security).

Algorithm: Encryption algorithm. The default value is **SunX509** for Sun Java JRE.

CipherSuites: Comma-separated list of cipher suites.

Protocols: Comma-separated list of SSL/TLS protocol names.

RESTInterface/[IPWhiteList, IPBlackList]

IPWhiteList: Comma-separated list of IP addresses that are authorized to use the REST interface. Wildcard (*****) characters are supported (for example, **192.168.*.***). The default value is the localhost loopback address (**127.0.0.1**)

IPBlackList: Comma-separated list of IP addresses that are prohibited from using the REST interface. Wildcard (*****) characters are supported.

RESTInterface/[EnableXMLFile]

EnableXMLFile: Boolean value that enables/disables the `/v2/servers/{server}/xml/{filename}` REST API (for example,

http://127.0.0.1:8087/v2/servers/_defaultServer_/xml/conf/Server). For security reasons, it's highly recommend that you keep this value set to the default value (**false**).

RESTInterface/[DocumentationServerEnable, DocumentationServerPort, DocumentationServerAuthenticationMethod]

DocumentationServerEnable: Boolean value that turns the REST API documentation for Wowza Streaming Engine on or off.

DocumentationServerPort: Port used to access the REST API documentation (**8089**).

DocumentationServerAuthenticationMethod: HTTP authentication method used to access the REST API documentation. This can be one of the following types: **none**, **basic**, or **digest**. The default value is **digest**.

Example URLs for accessing REST API documentation:

- http://localhost:8089
- http://*username*:*password*@localhost:8089

RESTInterface/Properties

Custom properties with name, type, and value, as defined by the user.

CommandInterface/HostPort

Wowza Streaming Engine command interface. This interface is used only to allow remote shutdown of the server. The entire **CommandInterface** section should be commented-out when a server is deployed in production.

ProcessorCount: Number of threads allocated to the **CommandInterface**: ${com.wowza.wms.TuningAuto}

IpAddress: IP address or domain name of the address that the **CommandInterface** listens to for incoming requests. If **IpAddress** is set to the wildcard (*) character, it tries to listen for incoming connections on all available network interfaces.

Port: 8083 is the command interface port used to shut down the server during testing. It can't be used when the **CommandInterface** section is commented-out, as it should be when a server is deployed in production

AdminInterface/ObjectList

List of objects made available through the Java Management Extensions (JMX) interface. For more information, see the article Use JConsole with Wowza Streaming Engine.

Stats/Enable

Interface that returns server statistics to administrators through Wowza Streaming Engine Manager or **HTTPProviders** on port 8086. These statistics reference the HTTP providers **serverinfo**, **connectioncounts**, and **connectioninfo**.

Enable: Boolean option that specifies if statistics are compiled and sent to the Wowza Streaming Engine Manager console or **HTTPProviders**.

JMXRemoteConfiguration/[*]

Java Management Extensions (JMX) interface configuration. For more information, see the article Use JConsole with Wowza Streaming Engine.

UserAgents

Values in HTTP requests that are interpreted as RTMPT requests. Any HTTP request header that includes a **UserAgents** value is considered to be an RTMPT request.

Streams/DefaultStreamPrefix

Specification of the streaming protocol default type. The default is **mp4**, but this type can be overridden in **IServerNotify2**, which is the server listener in **onServerConfigLoaded**. The **IServer** APIs that control the setting of this value are **IServer.setDefaultStreamNamePrefix(String streamNamePrefix)** and **String IServer.getDefaultStreamNamePrefix()**. The variable is also settable in the file **[install-dir]conf/Server.xml.ServerListeners**.

ServerListeners

Set of event handlers that allow developers to invoke server-side Java from JavaScript. **ServerListeners** can be used as a proxy to bypass client-side security. Configure server event listeners and virtual host (VHost) event listeners in the **Server.xml** file. Call these listeners when events occur at a server or VHost level.

VHostListeners

Set of event handlers that capture VHost events in order to extend server functionality. Server event listeners and VHost event listeners can be configured in the **Server.xml** file. These listeners are called when events occur at a server or VHost level. For the triggering

event to work, a VHost listener must be compiled, packaged into a **.jar** file, and placed in the **[install-dir]/lib** folder in the Wowza Streaming Engine installation. It can be invoked when an entry is added to the **<VHostListeners>** container in **[install-dir]/conf/Server.xml**.

HandlerThreadPool/PoolSize, TransportThreadPool/PoolSize

Maximum size of server-level threads in the handler and transport thread pools. The handler thread pool is used to process incoming messages. The transport thread pool is used to read/write data from the transport sockets. Server-level thread pools are only used if a VHost's thread pool size is set to **0**. This server-level thread pool is also used to process the **shutdown** command; therefore, it should never be set to a value less than **10**.

RTP/DatagramStartingPort

Lowest UDP port value assigned to incoming UDP streams. Ports are assigned starting with this value and are then incremented by 1. The most common value for RTSP/RTP-based servers is **6970**. If you plan to support RTSP/RTP, native RTP, or MPEG-TS streams, you should open UDP ports 6970-9999.

RTP/DatagramPortSharing

Boolean value that specifies port sharing options. If set to **true**, then UDP ports (both unicast and multicast) can be shared between Session Description Protocol (SDP) files and MPEG-TS streams or shared between application instances. Setting this value to **true** enables two SDP files to share RTP ports. For example, if you have a single video stream with multiple audio streams (each in a different language), then you can create two SDP files that share the single RTP video stream and refer to a unique audio stream. This also enables the same SDP file or MPEG-TS stream to be loaded by different application instances.

If set to **false**, UDP ports aren't allowed to be shared and an error is generated if an attempt is made to reuse a UDP port.

Manager

Container element for properties defined to be used by Wowza Streaming Engine Manager.

Properties

Typed name/value pairs. Properties are available in the Java API through the **IServer.getProperties()** interface. Custom properties are added to the collection returned by **IServer.getProperties()**.

MediaCache.xml

The **MediaCache.xml** configuration file is used to configure the Media Cache system in Wowza Streaming Engine. Media Cache can retrieve content from either a web server or HTTP (the server must support HTTP/1.1 byte range requests) or network attached file system or any similar device that's recognized as a disk by the operating system.

MediaCache

WriterThreadPool/PoolSize: Number of threads in the writer pool that are used to write media blocks to the caching system. This value should be set to twice the number of CPU cores on the machine.

ReadAheadThreadPool/PoolSize: Number of threads in the readahead pool that are used to read blocks from the cache source before they are requested. The readahead system keeps a steady flow of bytes moving from the source to the cache to avoid stuttering during playback. This value should be set to the number of CPU cores on the machine

MaxPendingWriteRequestSize: Number of bytes of memory that can be occupied by blocks waiting to be written to storage. Think of this storage area as a temporary memory-based cache. Values are specified in bytes and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

MaxPendingReadAheadRequestSize: Number of bytes of memory that can be occupied by blocks waiting to be written to the cache store. Values are specified in bytes and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

GCFrequency: Time, in milliseconds, between cache-purging/cache-pruning sessions. Based on time-to-live settings, items stored in the cache are purged when they haven't been used in a given period of time—maximum time-to-live, or if there's content waiting to enter the cache—minimum time-to-live. The default value is **10000** (10 seconds).

ContextMapperClass: Reserved for future use (leave blank).

AddFileExtensionIfNeeded: Indication of whether to include a file name extension. If **true**, and the stream name doesn't include a file name extension, then a file name extension is added. The value of this extension is based on the stream name prefix. Set this property to **false** to prevent the Media Cache from automatically adding file name extensions to cache items.

URLEscapeStreamNameSpaces: Boolean. If **true**, spaces in stream names are URL-escaped before being sent to the source HTTP server. The default value is **true**.

URLEscapeStreamNameAll: Boolean. If **true**, the entire stream name is URL-escaped before being sent to the source HTTP server. The default value is **false**.

OnStartReloadCache: Boolean. This value specifies whether to reload the cache when the server starts.

OnStartReloadCacheVerifySource: Boolean. This value specifies whether to verify the source when reloading the cache.

DebugLog: Boolean. When set to **true**, this turns on verbose logging for debugging.

MediaCacheStore

Name: Cache store name.

Description: Text description of the cache store.

Path: Path to the storage location. Always use forward slashes when defining paths, for example, **${com.wowza.wms.context.ServerConfigHome}/mediacache**. Cached files are stored in a two-tier directory structure.

MaxSize: Maximum size, in bytes of data. that's stored in this cache store. Values are specified in bytes, and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

Level1FolderCount: Number of first-level folders created to store cached items.

Level2FolderCount: Number of second-level folders created to store cached items.

FileCount: Number of items that can be stored in each level2 folder. For example, if the level1 folder count is set to **24**, and the level2 folder count is set to **24**, and the file count is set to **1000**, then 24x24x1000 or 576000 (the product of 24x24x1000) files can be stored in this cache store.

WriteRate: Maximum rate at which content can be written to this cache store. Throttling the write rate helps control the flow of content that enters the cache so that it doesn't overwhelm the file system, impeding the flow of content that's being served out of the cache. Values are expressed in bytes per second and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

WriteRateMaxBucketSize: Value that works in concert with **WriteRate** to control how the write rate is throttled. This value should be set to 6 times the **WriteRate**. Values are specified in bytes and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

WriteRateFillFrequency: Refresh rate, in milliseconds, of the write rate control mechanism. This value works in concert with **WriteRate** to control how the write rate is throttled. This value should be set to **100**.

Properties: Custom properties with name, value, and type that are defined by the user. Properties can be retrieved by using the **MediaCacheStore.getProperties()** method.

MediaCacheSource

Descriptions of sources of content that are made available to Wowza Streaming Engine. Settings are enumerated and used as follows:

Name: Name of the source, used for logging purposes. This name isn't used to control streaming or the addressing of the content.

Type: Identifies the Media Cache source type:

- **File**. For re-streaming files from network-attached file systems, including network file systems and any device that's recognized as a disk by the operating system.
- **HTTP**. For re-streaming files from HTTP-based servers that support HTTP/1.1 range requests.
- **AmazonS3**. For re-streaming files from an Amazon S3 bucket.
- **Azure**. For re-streaming content from a Microsoft Azure blob storage account.
- **GoogleCloudStorage**. For re-streaming content from a Google Cloud Storage bucket connected to a Google service account.

Description: Text description of the Media Cache source under examination.

BasePath and **Prefix**: Setting that works together with **Prefix** to control how content is mapped back to a source configuration. The **Prefix** is used to map the stream name to the source. To restream the content, the **Prefix** portion is replaced by the **BasePath** value. For example, if you set **Prefix** to **content1/** and **BasePath** to **http://**, then the stream name: **content1/mycoolvideo.flv** is retrieved from the following URL: **http://mycoolvideo.flv**.

BaseClass: Full class path to the MediaCacheSource implementation that's used to deliver the content.

- **File-based content**: com.wowza.wms.mediacache.impl.MediaCacheItemFileImpl
- **HTTP content**: com.wowza.wms.mediacache.impl.MediaCacheItemHTTPImpl

ReaderClass: Reserved for future use.

DefaultBlockSize: Size of the blocks in bytes of data that are read from the source to populate the cache. Values are specified in bytes and the following units are supported: K (kilobytes), M (megabytes), G (gigabytes), or T (terabytes).

MaxTimeToLive: Maximum time, in milliseconds, that an item will remain in the cache unused. The time-to-live counter starts when the last client to view the content stops viewing the content. If no other viewers view this content within the time-to-live window, the content is purged from the cache.

MinTimeToLive: Minimum time, in milliseconds, that an item will remain in the cache unused if there are items waiting to enter the cache, and the cache is full.

ReadAhead: Boolean value that controls the use of the readahead function. If **ReadAhead** is set to **true**, content blocks are pre-populated in the cache based on the **ReadAheadThreshold** percentage. If set to **false**, no readahead occurs.

ReadAheadThreshold: Percentage that controls when the request is made to read the next block. For example, if **ReadAheadThreshold** is set to 50 percent, then a successive content block is queued to be read when the current content block is read beyond the halfway mark.

Properties: Custom properties with name, value, and type, as defined by the user.

VHosts.xml

The **VHosts.xml** configuration file defines the virtual host (VHost) environments on the media server. By default, Wowza Streaming Engine ships with a single VHost environment named **_defaultVHost_**.

Tune.xml

The **Tune.xml** configuration file provides fine-tuning of settings for the media server. **Tune.xml** has three basic sections:

HeapSize

Defined settings for development and production environments, or the setting of an absolute value.

${com.wowza.wms.TuningHeapSizeProduction}: Assumption that Wowza Streaming Engine is the only application that's running on the server.

${com.wowza.wms.TuningHeapSizeDevelopment}: Assumption that Wowza Streaming Engine is sharing resources with other applications.

GarbageCollector

Defined settings that specify **GarbageCollector** options. **GarbageCollector** syntax includes the following options:

${com.wowza.wms.TuningGarbageCollectorConcurrentDefault}: Best concurrent garbage collector settings for your hardware. Uses a Java garbage collector that's designed for applications that prefer shorter GC pauses and that can share processor resources with the garbage collector while the application is running. There is a known issue when using the concurrent garbage collector with Java 11.

${com.wowza.wms.TuningGarbageCollectorG1Default}: Best G1 garbage collector settings for your hardware. Uses a Java garbage collector that's designed for low pause time, high-throughput applications. The G1 collector is a server-style garbage collector, targeted for multi-processor machines with large memories and is fully supported in Oracle JDK 7 Update 4 and later releases.

You can also specify a custom **GarbageCollector** setting directly as shown in the following example:

```
<GarbageCollector>-XX:+UseConcMarkSweepGC -XX:+UseParNewGC -
XX:NewSize=512m</GarbageCollector>)
```

VMOptions

Other Java command-line options.

-server: Appends the **server** option for maximum program execution speed for applications that are running in the server environment. When the **-server** option is appended, long-running server-side processes are optimized for that use case.

-Djava.net.preferIPv4Stack=true: Boolean. If IPv6 is available on the operating system, the underlying native socket is an IPv6 socket. When **true**, this setting allows Java applications to connect to, and accept connections from, both IPv4 and IPv6 hosts. If an application is configured to use only IPv4 sockets, then this property can be set to **true**. The implication of such a setting is that the application won't be able to communicate with IPv6 hosts.

-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath="${com.wowza.wms.AppHome}/logs": Dumps the heap to a file when **java.lang.OutOfMemoryError** is thrown. The path to the directory or file name that contains the heap dump is provided.

-Duser.language=en -Duser.country=US -Dfile.encoding=Cp1252: String values that set the language and country of the current virtual machine. These values specify the local language, the local country, and the default character encoding, or charset, for the Java virtual machine.

-verbose:gc -Xloggc:"${com.wowza.wms.AppHome}/logs/gc_${com.wowza.wms.StartupDateTime}.log" -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:+PrintHeapAtGC -XX:+PrintGCApplicationConcurrentTime -XX:+PrintGCApplicationStoppedTime: List of all currently set VMOptions (there should not be any newlines) as delineated by using the following switches:

- **VMOptions/VMOption/verbose**: Boolean. If you want to know when and where classes are loaded into the Java virtual machine, you can use the verbose option on the **java** command.
- **VMOptions/VMOption/Xloggc**: Boolean. Reports on each garbage collection event, as with **-verbose:gc**, but logs this data to a file. In addition to the information from **verbose:gc**, **Xloggc** logs each reported event preceded by the time, in seconds, since the first garbage-collection event.

> **Note:**
>
> Always use a local file system for storage of this file to avoid stalling the Java virtual machine (due to network latency). If you're working with a full file system, this file may be truncated and logging will continue in the truncated file. This option overrides **verbose:gc**.

- **VMOptions/VMOption/PrintGCDetails**: Boolean. This option will print additional detail at garbage collection.
- **VMOptions/VMOption/PrintGCTimeStamps**: Boolean. This option prints timestamps at garbage collection.
- **VMOptions/VMOption/PrintHeapAtGC**: Boolean. This option prints the heap layout before and after each garbage collection.
- **VMOptions/VMOption/PrintGCApplicationConcurrentTime**: Boolean. This option prints the length of time that the application has been running.
- **VMOptions/VMOption/PrintGCApplicationStoppedTime**: Boolean. This option prints the length of time that the application has been stopped.

log4j.properties

The **log4j.properties** file is used to configure logging for Wowza Streaming Engine. Wowza Streaming Engine uses the Java-based log4j logging system. By default, the server is configured to log basic information to the console window and detailed information in W3C Extended Common Log Format (ECLF) to log files. For more information about how to configure the logging system, see any of the articles listed on the Wowza Streaming Engine Logging documentation landing page.

Chapter 3

Virtual Host Configuration Files

StartupStreams.xml

The **StartupStreams.xml** configuration file lists the streams to start when a virtual host (VHost) starts. Streams must be valid **MediaCaster** types: **rtp**, **rtp-record**, **shoutcast**, **shoutcast-record**, or **liverepeater**.

StartupStream/Application

Application name and instance name to use when starting the stream in the form **[application]/[appInstance]**. If **[appInstance]** is omitted, the default application name **_definst_** is used.

StartupStream/StreamName

Name of the stream to start up. If needed, this value should include a stream name prefix.

StartupStream/MediaCasterType

MediaCaster type name used to start the stream.

VHost.xml

The **VHost.xml** configuration file defines the settings used to configure a virtual host (VHost). Wowza Streaming Engine can serve multiple users from separate virtual hosting environments. Each VHost environment has its own set of configuration files, application folders, and log files.

HostPortList/HostPort

List of IP addresses and TCP ports that Wowza Streaming Engine will bind to for incoming and outgoing streaming connections. Wowza Streaming Engine can be configured for any number of TCP ports. **HostPorts** are used to stream RTMP, RTSP, and HTTP. A **HostPort** can be configured to use Secure Sockets Layer (SSL) encryption. HTTP provider configuration is done on a per-**HostPort** basis.

HostPortList/HostPort/ProcessorCount

Number of threads allocated to use to service connections. For more information about recommended values to use based on server resources, see Tune Wowza Streaming Engine for optimal performance.

HostPortList/HostPort/[IpAddress, Port]

IpAddress: IP address or domain name of the address that Wowza Streaming Engine will listen to for incoming requests. If **IpAddress** is set to the wildcard (*****) character, the server will try to listen for incoming connections on all available network interfaces. **Port** is a comma-separated list of ports.

HostPortList/HostPort/HTTPIdent2Response

Utility class for parsing and converting host-port information from commands. Represents a list of host-port pairs.

HostPortList/HostPort/SocketConfiguration/[*]

Detailed socket connection configuration that's created by this **HostPort** definition at runtime. You can use these settings to tune the performance of the socket connections that are used to send data into and out of Wowza Streaming Engine. **SendBufferSize**,

ReceiveBufferSize, and **ReadBufferSize** are the most important settings in this group. They define the size of the memory buffers that are used during data transfer over the socket connection. Values of **0** for **SendBufferSize** and **ReceiveBufferSize** instruct Wowza Streaming Engine to use the operating system default values for these settings. For operating systems that support it, TCP auto-tuning at the kernel level may be used to set the buffer sizes dynamically for individual connections. For more information about recommended values to use based on server resources, see Tune Wowza Streaming Engine for optimal performance.

The **ReuseAddress** and **KeepAlive** settings should both be set to **true**. They are only provided for completeness.

The **AcceptorBackLog** setting controls the maximum number of TCP connection requests that can be pending before new connection requests are refused. Wowza Streaming Engine will respond to TCP connection requests as quickly as possible. TCP connection requests should not be set to a value of less than **50**. Setting this value to **-1** allows the operating system to control the maximum number of pending TCP connection requests.

Note

Using the "automatic" **-1** setting value isn't always the best decision. When interpreting a **-1** **AcceptorBacklog** value, some platforms will set a very small number as the maximum possible TCP connection requests and this can greatly increase connection times.

HostPortList/HostPort/HTTPStreamerAdapterIDs

Comma-separated list of **HTTPStreamers** that this **HostPort** will include when processing HTTP requests. **HTTPStreamerAdapterIDs** can contain none, one, or more of the following values (separated by commas): **cupertinostreaming** (Apple HLS), **smoothstreaming**, **sanjosestreaming** (Adobe HDS), **dvrchunkstreaming**, **mpegdashstreaming**.

HostPortList/HostPort/HTTPProviders

List of HTTP providers that this **HostPort** includes when processing HTTP requests. For more information, see Use HTTP providers with the Wowza Streaming Engine Java API.

HostPortList/HostPort/HTTPProviders/HTTPProvider

BaseClass: Base class of the **HTTPProvider** class, such as **com.wowza.wms.http.HTTPServerInfoXML**.

RequestFilters: HTTP request type limiter. To restrict the types of HTTP requests your Web server will process, configure the server to analyze specific criteria for each incoming request as specified by files such as **clientaccesspolicy.xml** and **crossdomain.xml**.

AuthenticationMethod: Authentication method specifier such as **basic**, **admin-digest**, **digest**, **block**, **admin-basic**, **admin-block**, or **none**.

HostPortList/HostPort/SSLConfig/[*]

Secure Sockets Layer (SSL) configuration for a given **HostPort**.

KeyStorePath. Full path to the keystore file.

KeyStorePassword. Keystore password.

KeyStoreType. Keystore type. The default value is **JKS** for Sun Java JRE.

SSLProtocol. Cryptographic protocol. The default value is **TLS** (Transport Layer Security).

Algorithm. Encryption algorithm. The default value is **SunX509** for Sun Java JRE.

CipherSuites: Comma-separated list of cipher suites.

Protocols: Comma-separated list of SSL/TLS protocol names.

HTTPStreamerAdapters

List of HTTP Streamers that are enabled for **HostPort** entries. All HTTP Streamers listed here can be streamed by using the **HostPort** IP address and port combination. If you want to disable a particular HTTP streaming protocol, you can remove it from this list.

ID: Identity that's one of the following options: **cupertinostreaming**, **sanjosestreaming**, **smoothstreaming**, **mpegdashstreaming**, **tsstreaming**, **webmstreaming**, or **dvrchunkstreaming**. These options represent the HTTP streaming protocols and well as DVR streaming from origin to edge in a Wowza live stream repeater configuration.

Name: Name for one of the following options: **cupertinostreaming**, **sanjosestreaming**, **smoothstreaming**, **mpegdashstreaming**, **tsstreaming**, **webmstreaming**, or **dvrchunkstreaming**.

Properties: Custom properties with path, name, type, and value, as defined by the user.

HandlerThreadPool/PoolSize, TransportThreadPool/PoolSize

Maximum size of the virtual host-level threads in the handler and transport thread pools. The handler thread pool is used to process incoming messages while the transport thread pool is used to read/write data from the transport sockets. If the pool size is set to **0** for a given thread pool type, the server-level thread pool of the same type is used for this virtual host. For more information about recommended values to use based on server resources, see Tune Wowza Streaming Engine for optimal performance.

IdleWorkers/[WorkerCount, CheckFrequency, MinimumWaitTime]

WorkerCount controls the number of threads being used to generate idle events. **CheckFrequency** is the time, in milliseconds, between system checks that monitor if a client has been idle for **Client/IdleFrequency**. The **CheckFrequency** value should be one-fourth or less of the **Client/IdleFrequency** value. For more information about recommended values to use based on server resources, see Tune Wowza Streaming Engine for optimal performance. **MinimumWaitTime** specifies an interval in seconds that the server will wait.

NetConnections/[ProcessorCount, IdleFrequency]

Settings used to tune connections made between Wowza Streaming Engine servers, for example, when using the live stream repeater. **ProcessorCount** is the number of threads allocated to connections. Increasing the **IdleFrequency** from the default 250 milliseconds will give the CPU a little more time so it won't be working as hard.

NetConnections/[SocketConfiguration]

ReuseAddress: Boolean. This value determines if the address of the incoming RTP datagram can be reused.

ReceiveBufferSize: Size in bytes of the buffer that receives incoming data.

ReadBufferSize: Size of the memory buffer that contains read data.

SendBufferSize: Size of the memory buffer that contains data to be sent.

Note

Values of **0** for **SendBufferSize** and **ReadBufferSize** configure Wowza Streaming Engine to use the operating system default values for these settings.

KeepAlive: Time, in milliseconds, that a connection remains open when no data is being transferred.

AcceptorBacklog: Number of connections allowed in backup queue.

MediaCasters/ProcessorCount

Number of threads allocated to MediaCasters: ${com.wowza.wms.TuningAuto}

MediaCasters/SocketConfiguration

ReuseAddress: Boolean. This value determines if the address of the incoming RTP datagram can be reused.

ReceiveBufferSize: Size in bytes of the buffer that receives incoming data.

ReadBufferSize: Size of the memory buffer that contains read data.

SendBufferSize: Size of the memory buffer that contains data to be sent.

Note

Values of **0** for **SendBufferSize** and **ReadBufferSize** configure Wowza Streaming Engine to use the operating system default values for these settings.

KeepAlive: Time, in milliseconds, that a connection remains open when no data is being transferred.

ConnectionTimeout: The time (in milliseconds) that this MediaCaster will wait when connecting to a remote service.

LiveStreamTranscoders/MaximumConcurrentTranscodes

Maximum number of concurrent live source streams that are transcoded at any point in time. This setting is useful if you licensed Wowza Streaming Engine with a license that enables an unlimited number of concurrent transcodes, and you want to limit the number of concurrent transcodes due to CPU/hardware limitations or for billing purposes. A value of **0** allows an unlimited number of concurrent transcodes.

HTTPTunnel/KeepAliveTimeout

Keepalive time, in milliseconds, for RTMPT, RTMPTE, and RTMPS connections.

Client/[ClientTimeout, IdleFrequency]

ClientTimeout, **IdleFrequency**: Respectively, the length of time, in milliseconds, that the server will wait before shutting down an unresponsive client connection and the length of time, in milliseconds, between idle events. For basic video on demand (VOD) streaming, a value of **250** provides the best reliability-versus-performance ratio. For live streaming, a value between **125** and **250** is more desirable. Higher values will increase the frequency at which media data is sent to Adobe Flash clients. If you adjust this value, be sure to also adjust **IdleWorkers/CheckFrequency** to a value that's less than or equal to one-fourth of this value. For example, if the value of **ClientTimeout** and **IdleFrequency** is **200**, then the value of **IdleWorkers/CheckFrequency** should be **50** or less.

RTP/IdleFrequency

Time, in milliseconds, between idle events for RTP connections.

RTP/DatagramConfiguration/[Incoming, Outgoing]/[*]

Datagram socket configuration for incoming and outgoing RTP connections.

ReuseAddress. Boolean value that determines if the address of the incoming/outgoing RTP datagram can be reused.

ReceiveBufferSize. Size, in bytes, of the incoming UDP buffer.

SendBufferSize. Size, in bytes, of the outgoing UDP buffer.

MulticastTimeout. Timeout value, in milliseconds, for multicast polling.

DatagramMaximumPacketSize. Maximum size, in bytes, for a single incoming UDP packet.

RTP/[*]/ProcessorCount

Number of threads allocated to incoming and outgoing unicast and multicast UDP streams.

HTTPProvider

KeepAliveTimeout: Time, in milliseconds, that the TCP session is maintained after a response is sent.

KillConnectionTimeout: Time, in milliseconds, that the TCP session validation code waits before a TCP session is terminated because of inactivity.

SlowConnectionBitrate: Estimates the time, in bits-per-second, that it takes to flush data from the TCP send buffer when determining the **KeepAliveTimeout** time.

IdleFrequency: Time, in milliseconds, between TCP idle events.

WebSocket

MaximumMessageSize: Maximum size, in bytes, of a single WebSocket message. If any message is larger than this value, the TCP session is terminated to guard against memory run-up. If set to **0**, this property isn't used.

MaskOutgoingMessages: Boolean value that determines if outgoing messages are masked. Note that most modern web browsers don't accept masked messages.

IdleFrequency: Time, in milliseconds, between TCP idle events.

ValidationFrequency: Time, in milliseconds, between sending messages from the server to the browser to validate that the TCP session is still active.

MaximumPendingWriteBytes: Maximum number of bytes queued to be written. If this value is exceeded, the TCP session is terminated to guard against memory run-up. If set to **0**, this property isn't used.

PingTimeout: Time, in milliseconds, the WebSocket waits for a response to a ping request.

Application/ApplicationTimeout

Length of time, in milliseconds, that the server will wait before shutting down an application to which no clients are connected. A value of **0** keeps applications running until the virtual host is shut down. If this value isn't provided, the value set in the **VHost.xml** file is used.

Application/PingTimeout

Time, in milliseconds, that the server will wait for a ping response from the client. The ping mechanism that's used to validate a client connection is an RTMP internal ping, not an ICMP ping. If **PingTimeout** is set to **0**, the server will wait indefinitely.

Application/UnidentifiedSessionTimeout

Time, in milliseconds, that the server will wait for an unidentified client session before disconnecting the application.

Application/ValidationFrequency

Time, in milliseconds, that the server will wait during server-to-client validation. Validation only occurs if the client stops sending data to the server. Validation is done by sending a ping request from the server to the client. If **ValidationFrequency** is set to **0**, the server won't validate client connections.

Application/MaximumPendingWriteBytes

Maximum number of bytes that are allowed to be queued and remain waiting to be sent. If this value is exceeded on an individual client connection, that connection is terminated. If **MaximumPendingWriteBytes** is set to **0**, there is no maximum value.

Application/MaximumSetBufferTime

Maximum buffer time allowed on the client side, in milliseconds. A value for **MaximumSetBufferTime** of **0** removes the buffer time limit. This property can be used to deter stream rippers that might connect to Wowza Streaming Engine and attempt to set a very large buffer time in order to steal content. Unlimited buffer time can lead to server-side memory run-up.

StartStartupStreams

Boolean value that indicates system action upon startup streams. When set to **true**, streams that are defined in **StartupStreams.xml** are instantiated at server startup. When set to **false**, streams aren't started at server startup.

Manager/TestPlayer/[IpAddress, Port, SSLEnable]

IpAddress: IP address or server name that the **TestPlayer** should use to connect to the server. Default: **${com.wowza.wms.HostPort.IpAddress}**

Port: Port that the **TestPlayer** should use to connect to the server. Default: **${com.wowza.wms.HostPort.FirstStreamingPort}**

SSLEnable: Enables the test player to use a VHost port with SSL enabled. Sets the stream URL protocol to **https:** or **rtmps:** The default value is ${com.wowza.wms.HostPort.SSLEnable}, which will query the hostport in **VHost.xml** and set to **true** or **false** depending on its value.

Note

When Wowza StreamLock™ AddOn is configured, **IpAddress** must be set to the StreamLock certificate name, for example: `<IpAddress>52f51948efc92.streamlock.net</IpAddress>`

If a nonstandard SSL port is used, set it explicitly to the port, for example: `<Port>9443</Port>`. If you have enabled StreamLock for the default streaming port (1935), keep the default **Port** value.

The values **true**/**false** for **SSLEnable** determine which URL prefix is used in test player URLs: **true** (rtmps or https) and **false** (rtmp or http).

When multiple streaming ports are defined for a VHost, the default port (`${com.wowza.wms.HostPort.FirstStreamingPort}`) can be over-ridden.

Amazon EC2 instances use private and public DNS values as IP addresses. EC2 private/public IP address usage with the StreamLock feature requires the private DNS value or the wildcard (*****) character to be used as the IP address in order for EC2 AMIs with StreamLock configurations to work. Bind failures occur if the EC2 public DNS value is used as the IP address. You must also restart the server to update StreamLock changes. Restarting the VHost will not update StreamLock changes.

Properties

Typed name/value pairs. All virtual host properties are copied to virtual hosts upon creation. These properties are available in the Java API through the **IVHost.getProperties()** interface.

Chapter 4

Application Configuration Files

Application.xml

The **Application.xml** configuration file defines the configuration settings for an application. An application is a contextual state/environment for media streaming that specifies a set of operating properties within Wowza Streaming Engine. Users can create as many applications as needed for live streaming, video on demand, transcoding, nDVR, or re-streaming from other servers and live sources.

Name, AppType, and Description

Name: The name of the application. Required.

AppType: The type of the application, such as **live**, **vod**, **livehttporigin**, **vodhttporigin**, **liveedge**, **vodedge**. Required.

Description: A text description of the application. Optional.

ApplicationTimeout

Length of time, in milliseconds, that the server will wait before shutting down an application to which no clients are connected. A **Timeout** value of **0** keeps applications running until the virtual host is shut down. If this value is not provided (or the section is commented out), the value set in the **VHost.xml** file is used.

PingTimeout

Length of time, in milliseconds, that the server will wait for a ping response from the client. The ping mechanism that's used to validate a client connection is an RTMP internal ping, not an ICMP ping. If **PingTimeout** is set to **0**, the server will wait indefinitely. If this value isn't provided (or the section is commented out), the value set in the **VHost.xml** file is used.

ValidationFrequency

Length of time, in milliseconds, that the server will wait during server-to-client validation. Validation only occurs if the client stops sending data to the server. Validation is done by sending a ping request from the server to the client. If **ValidationFrequency** is **0**, the server won't validate client connections. If this value isn't provided (or the section is commented out), the value set in the **VHost.xml** file is used.

MaximumPendingWriteBytes

Maximum number of bytes that are allowed to be queued and remain waiting to be sent. If this value is exceeded for an individual client connection, then the connection is terminated. If **MaximumPendingWriteBytes** is set to **0**, there is no maximum value. If this value isn't provided (or the section is commented out), the value set in the **VHost.xml** file is used.

MaximumSetBufferTime

Maximum number of milliseconds honored server-side for client-side calls to **NetStream.setBufferTime(seconds)**. To turn off this check, set the value of **MaximumSetBufferTime** to **0**. The default value is **60000** (60 seconds). This setting is used to protect against spoofing threats (such as those posed by Replay Media Catcher and Grab Pro), which can set a very large client-side buffer to trick a server into sending all the media data at once. This can cause the server to consume a large amount of Java heap memory.

MaximumStorageDirDepth

Maximum number of subfolders that are allowed in any storage path. This setting helps to protect against symbolic link loops.

Connections/AutoAccept

Setting that determines if the application automatically accepts Adobe Flash client playback connection requests. If set to **true**, all connection requests are accepted automatically. If set to **false**, the application must make a server-side call to **client.acceptConnection()** to accept a connection request.

Connections/AllowDomains

Comma-delimited list of domain names or IP addresses for which client connections are accepted. The domain names or IP addresses that are specified represent the domain name

or IP address of an Adobe Flash SWF file that connects to Wowza Streaming Engine or the IP address of a client that connects to Wowza Streaming Engine. If no value is specified, then connections from all domains or IP addresses are accepted.

For example, if you have the SWF file **http://www.mycompany.com/flash/myflashmovie.swf**, you can set **AllowDomains** to **www.mycompany.com** in order to configure the Wowza Streaming Engine instance so that only clients from the **mycompany.com** domain can access the server. You can also add an IP address (or IP address wildcard) to accept all connections from a particular IP address. You might filter based on IP address when you're working with a client-side live source that doesn't provide a valid referrer.

You can use the wildcard character (*****) to match partial domain names or IP addresses. For example, if you want to match all domain names that end with **mycompany.com**, you would specify the domain name ***.mycompany.com**.

Allow-domains processing occurs just before the **onConnect** event method. Therefore, if you want to provide finer-grained access control to your server, you can override the **onConnect** event handler in a custom module and provide your own filtering mechanism.

Streams/StreamType

Name of the default stream type for this application. For more information about stream types, see "Stream Types" in the Wowza Streaming Engine User Guide.

Streams/StorageDir, KeyDir, SharedObjects/StorageDir

Streams/StorageDir: Full path to the directory where the application reads and writes media files. **KeyDir** is the directory where the Cupertino streaming (Apple HLS) AES-128 encryption keys are stored. **SharedObjects/StorageDir** is the full path to the directory where the application reads and writes remotely stored object data. If no values are specified, an application uses the following directories:

```
%WMSCONFIG_HOME%/applications/[application]/streams/[appinstance]
%WMSCONFIG_HOME%/applications/[application]/sharedobjects/[appinstance]
%WMSCONFIG_HOME%/applications/[application]/keys/[appinstance]

%WMSCONFIG_HOME%      the value of the environment variable WMSCONFIG_HOME
[application]         the name of the application
[appinstance]         the name of the application instance
```

The following variables are supported:

Variable Name	Description
${com.wowza.wms.AppHome}	Application home directory
${com.wowza.wms.ConfigHome}	Configuration home directory
${com.wowza.wms.context.VHost}	Virtual host name
${com.wowza.wms.context.VHostConfigHome}	Virtual host config directory
${com.wowza.wms.context.Application}	Application name
${com.wowza.wms.context.ApplicationInstance}	Application instance name

Streams/LiveStreamPacketizers

HTTP streaming packetization schemes to use for live source streams. Live stream packetization is done to make a stream available for HTTP streaming to the Apple iPhone and iPod touch, Adobe Flash Player, Microsoft Silverlight, and MPEG-DASH players. It is also done to enable DVR for a live stream. **LiveStreamPacketizers** can contain none, one, or more of the following values (separated by commas):

LiveStreamPacketizers	Description
cupertinostreamingpacketizer	"Cupertino" HLS streaming using MPEG-TS container (iOS playback)
cmafstreamingpacketizer	HLS and MPEG-DASH streaming using fMP4 container
smoothstreamingpacketizer	Microsoft Silverlight playback
sanjosestreamingpacketizer	"San Jose" streaming (Adobe Flash playback)
mpegdashstreamingpacketizer	MPEG-DASH streaming using fMP4 container
cupertinostreamingrepeater	Cupertino: Live stream repeater for iOS devices
smoothstreamingrepeater	Cupertino: Live stream repeater for Microsoft Silverlight
sanjosestreamingrepeater	San Jose: Live stream repeater for Adobe Flash
mpegdashstreamingrepeater	MPEG-DASH: Live streaming repeater for MPEG-DASH players
dvrstreamingpacketizer	Wowza nDVR: Streaming
dvrstreamingrepeater	Wowza nDVR: Live stream repeater

Streams/Properties

Defined properties and name/value pairs for instances of **Streams**.

Transcoder/LiveStreamTranscoder

Name of the default Transcoder handler for this application. If set to **transcoder**, the Transcoder is enabled for this application. If no value is specified, live source streams aren't transcoded.

Transcoder/Templates

Comma-separated list of template names to search when matching a live source stream to a Transcoder template. If the Transcoder is enabled, each new live stream that's published to the application can be transcoded. Wowza Streaming Engine will search the **Templates** list for the first template file that exists. After it finds a template, it uses that template to transcode the stream. If no matches are found, the stream won't be transcoded. The default value for this setting is:

```
${SourceStreamName}.xml, transrate.xml
```

The first item in the template name list uses the variable **${SourceStreamName}**. If there's a Transcoder template in the template directory with the name **[stream-name].xml** (where **[stream-name]** is the name of the live source stream), then that template is used. If this file doesn't exist, the **transrate.xml** template is used (if it exists). If neither template exists, then the stream won't be transcoded. Only the variable **${SourceStreamName}** is supported.

Transcoder/ProfileDir

Environment variable that specifies the path to the Transcoder profiles:
{com.wowza.wms.context.VHostConfigHome}/transcoder/profiles

Transcoder/TemplateDir

Full path to the directory where the application looks for templates that are used to control the Transcoder. By default, Wowza Streaming Engine looks for templates in the **[install-dir]/transcoder/templates** folder, for example, **${com.wowza.wms.context.VHostConfigHome}/transcoder/templates**. You can also define a per-application templates folder by using path variables. For example, if you want to set up a per-application templates folder, specify the following path:

```
${com.wowza.wms.context.VHostConfigHome}/conf/${com.wowza.wms.context.Appli
cation}/profiles
```

The following variables are supported:

Variable Name	Description
${com.wowza.wms.AppHome}	Application home directory
${com.wowza.wms.ConfigHome}	Configuration home directory
${com.wowza.wms.context.VHost}	Virtual host name
${com.wowza.wms.context.VHostConfigHome}	Virtual host config directory
${com.wowza.wms.context.Application}	Application name
${com.wowza.wms.context.ApplicationInstance}	Application instance name

Transcoder/Properties

Name/value pairs defined by the user for the Transcoder.

DVR/Recorders

Name of the default DVR recorder for this application.

DVR/Store

As a single server or as an origin, set the **Store** to **dvrfilestorage**. **Store** should be empty for edge servers.

DVR/WindowDuration

Duration, in seconds, of material available for DVR playback. The default value of **0** denotes that there's no DVR window (all data is available for DVR playback).

DVR/StorageDir

Full path to the directory where the application reads and writes Wowza nDVR data.

The following variables are supported:

Variable Name	Description
${com.wowza.wms.AppHome}	Application home directory
${com.wowza.wms.ConfigHome}	Configuration home directory
${com.wowza.wms.context.VHost}	Virtual host name
${com.wowza.wms.context.VHostConfigHome}	Virtual host config directory
${com.wowza.wms.context.Application}	Application name
${com.wowza.wms.context.ApplicationInstance}	Application instance name

DVR/ArchiveStrategy

Value that tells the DVR store what to do with an old stream when a new stream of the same app instance and stream name starts. The default value is **append**. **ArchiveStrategy** can contain only one of the following values:

ArchiveStrategy	Description
append	Append the new stream information to the end of the previous store.
delete	Delete the old DVR store and start a new one.
version	Create a new version of the DVR store.

DVR/Properties

Defined DVR properties and name/value pairs for this application.

TimedText/VODTimedTextProviders

Comma-separated list of video-on-demand (VOD) caption providers. These caption providers are declared in **TimedTextProviders.xml**.

Properties

Additional configuration settings and name/value pairs for fine tuning **TimedText**.

HTTPStreamers

Streaming protocols supported by Wowza Streaming Engine. HTTP streaming—which supported by Apple iOS devices, Adobe Flash, Microsoft Silverlight, and MPEG-DASH—also allows DVR streaming to repeat DVR audio and video content from the origin to the edge. **HTTPStreamers** can contain none, one, or more of the following values (separated by commas):

HTTPStreamers	Description
cupertinostreaming	Cupertino: HTTP streaming to iOS devices
smoothstreaming	Smooth: HTTP streaming to Microsoft Silverlight
sanjosestreaming	San Jose: HTTP streaming to Adobe Flash
mpegdashstreaming	MPEG-DASH: HTTP streaming to MPEG-DASH players
dvrchunkstreaming	DVR: Enable streaming from origin to edge

MediaCacheSourceList

Specifies which of the Media Cache Sources that are configured on the system can be accessed by a VOD edge application. The wildcard (*****) character means that all Media Cache sources configured on the system can be accessed by the application. Selected Media Cache sources can be specified as a comma-separated list. Media Cache Sources are HTTP-based servers and network-attached file systems that provide file-based content to the Media Cache system for re-streaming. For example, a VOD application running on an Amazon EC2 instance could include amazons3 and/or dvrorigin Media Cache Sources.

Client/IdleFrequency

Time, in milliseconds, between idle events. Idle events represent the heartbeat of streaming for Adobe Flash and RTSP/RTP streaming. The idle frequency represents how often new data is sent to the player. If this value is set to **-1**, then the value specified in the **VHost.xml** file is used.

Client/Access/[*]

Settings that control the default access that an Adobe Flash client connection has to assets associated with a particular Wowza Streaming Engine application. An individual client's access can be modified through the Java API. This is most commonly done in the **onConnect** or **onConnectAccept** event handler. To control access, each of these settings compares the asset name (stream name or shared object name) to a comma-delimited list of names. If any

part of the asset name matches one of the elements in the list, then the requested access is granted. The values are case-sensitive. If no parameter value is specified, then access is denied to all clients. If the parameter value is set to the wildcard (*****) character, then access is granted to all clients. For example, if **StreamReadAccess** is set to **testa/testb;testc**, then the following stream name would be granted the following access:

Stream Name	Access Status
testc	Granted Access
testc/test	Granted Access
testC/test	Denied Access (incorrect case)
testa/testb	Granted Access
testa/testb123	Granted Access
testa/testb/file123	Granted Access
testa/test	Denied Access (incomplete match)

StreamReadAccess: Setting controls the access to view or listen to a **NetStream** object.

StreamWriteAccess: Setting controls the access to write or publish to a **NetStream** object.

StreamAudioSampleAccess: Setting controls the access to call **SoundMixer.computeSpectrum()** to grab the waveform data of a **NetStream** object.

StreamVideoSampleAccess: Setting controls the access to call **BitmapData.draw()** to take a snapshot of a **NetStream** object.

SharedObjectReadAccess: Setting controls the access to read values from a **RemoteSharedObject**.

SharedObjectWriteAccess: Setting controls the access to write values to a **RemoteSharedObject**.

RTP/Authentication/[PublishMethod, PlayMethod]

Authentication method used to secure RTSP connections to Wowza Streaming Engine. **PublishMethod** is for source connections and **PlayMethod** is for playback connections. Authentication methods are defined and configured in **Authentication.xml**. By default, there are three authentication methods:

- **none**: There is no authentication.
- **basic**: User name and password are sent in cleartext.
- **digest**: Password is hashed using MD5 and is never sent in cleartext over the network.

User names and passwords are stored in the **conf/publish.password** file. The **publish.password** file contains one line per user formatted as `[user name][space][password]`. The authentication method can also be set at the virtual host level in **VHost.xml**.

RTP/[AVSyncMethod, MaxRTCPWaitTime]

Settings that control how Wowza Streaming Engine synchronizes audio and video channels when receiving an RTP stream.

AVSyncMethod configures the methodology used to synchronize the audio and video channels using one of the following values:

- **senderreport**: Uses the Sender Report (SR) packets that are sent over the Real Time Control Protocol (RTCP) channel. This is the default value.
- **rtptimecode**: Assumes that RTP timecodes are absolute timecode values.
- **systemclock**: Synchronizes based on the system clock.

MaxRTCPWaitTime is the maximum period of time, in milliseconds, that the Wowza Streaming Engine instance will wait to receive an SR packet over the RTCP channel. If no SR packets are received within this time, the server defaults to using the **rtptimecode** method.

RTP/IdleFrequency

Length of time, in milliseconds, between RTP idle events. Idle events are used to send new media data and events to an RTP or MPEG-TS sessions.

RTP/RTSPSessionTimeout

Elapsed time, in milliseconds, after which an idle RTSP session is determined to be stale and is disconnected. To turn off idle disconnect, set this value to **0**. Wowza Streaming Engine monitors all RTSP sessions and looks for periodic RTSP messages or RTCP receiver packets over RTP. If the server hasn't received messages during the session timeout period, the session is disconnected.

RTP/RTSPMaximumPendingWriteBytes

Maximum number of bytes that can wait to be written to an RTSP session. If an RTSP session has more bytes waiting to be written than are specified by this setting, the session is disconnected. Wowza Streaming Engine monitors the RTSP/TCP connection and watches the number of bytes that are waiting to be delivered. If the number of pending bytes exceeds the

specified value, then the session is determined to be inactive and the session is disconnected. To turn off maximum pending write byte monitoring, set the **RTP/RTSPMaximumPendingWriteBytes** value to **0**.

RTP/[RTSPBindIpAddress, RTSPConnectionIPAddress, RTSPOriginIpAddress]

Setting that controls the IP addresses that are exchanged and used during RTSP/RTP port and IP address negotiation when the RTP portion of the stream is delivered over UDP. The **RTSPBindIpAddress** property is the IP address that Wowza Streaming Engine binds to when delivering RTP packets over UDP. If the server is using network address translation (NAT) routing, then this address should be set to the internal IP address of the network interface that's mapped to the external IP address. If NAT isn't being used, then this value should be set to the external IP address of the server. The **RTSPConnectionIpAddress** and **RTSPOriginIpAddress** are the IP addresses that are exchanged as part of the Session Description Protocol (SDP) data. The **RTSPConnectionIpAddress** value is the IP address specified in the `(c=)` line of the SDP data and the **RTSPOriginIpAddress** value is the IP address specified in the `(o-)` line of the SDP data. These two values should be set to the external IP address of the server.

RTP/IncomingDatagramPortRanges

UDP port ranges that this application can use for stream ingestion. Single ports and port ranges can be defined. For example, the following entry enables port **10000** and ports **20000** through **20004**:

```
<IncomingDatagramPortRanges>10000, 20000-20004</IncomingDatagramPortRanges>
```

This setting only affects stream ingestion when using Session Description Protocol (SDP) files or MPEG-TS udp:// URLs to pull streams. The values contained in **IncomingDatagramPortRanges** don't affect UDP ports that are dynamically assigned during RTSP port negotiation.

RTP/Properties

Supplemental configuration settings for tuning RTP.

MediaCaster/RTP/RTSP/[RTPTransportMode]

RTSP flavor that's used to re-stream an RTSP/RTP source, such as an IP camera. If set to **interleave**, Wowza Streaming Engine uses RTSP/RTP interleaved (RTP over TCP) to connect

to RTSP sources. If set to **udp**, Wowza Streaming Engine uses RTSP/RTP in non-interleaved mode (RTP over UDP) to connect to RTSP sources.

MediaCaster/[StreamValidator]

Enable: Boolean. If set to **true**, then the **StreamValidator** is active.

ResetNameGroups: Boolean. If set to **true**, then when a stream is reset and it belongs to a **MediaStreamNameGroup**, all streams in the group are reset. If **false**, only the unhealthy stream is reset.

StreamStartTimeout: Setting that controls the timeout, in milliseconds, for the first packet when monitoring incoming packets (audio, video, data) to be sure packets continue to flow from the live source to stream.

StreamTimeout: Setting that controls the timeout, in milliseconds, for packets after the first packet. The stream type refers to a catch-all of any packet of any type (audio, video, data). If any of these values are set to **0**, the test is turned off.

VideoStartTimeout: Setting that controls the timeout, in milliseconds, for the first packet when monitoring incoming video packets to be sure packets continue to flow from the live source to stream.

VideoTimeout: Setting that controls the timeout, in milliseconds, for video packets after the first packet.

AudioStartTimeout: Setting that controls the timeout, in milliseconds, for the first audio packet when monitoring incoming packets.

AudioTimeout: Setting that controls the timeout, in milliseconds, for audio packets after the first packet.

VideoTCToleranceEnable: Boolean value that monitors video timecode jumps when set to **true**.

VideoTCPosTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous video packets. If **videoTCPosTolerance** and **videoTCNegTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous video packet must fall within 3000 and -500 milliseconds.

VideoTCNegTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous video packets. If **videoTCNegTolerance** and **videoTCPosTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous video packet must fall within 3000 and -500 milliseconds.

AudioTCToleranceEnable: Boolean value that monitors audio timecode jumps when set to **true**.

AudioTCPosTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous audio packets. If **AudioTCPosTolerance** and **AudioTCNegTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous audio packet must fall within 3000 and -500 milliseconds.

AudioTCNegTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous audio packets. If **AudioTCNegTolerance** and **AudioTCPosTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous audio packet must fall within 3000 and -500 milliseconds.

DataTCToleranceEnable: Boolean value that monitors timecode jumps when set to **true**.

DataTCPosTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous data packets. If **DataTCPosTolerance** and **DataTCNegTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous data packet must fall within 3000 and -500 milliseconds.

DataTCNegTolerance: Value that defines the allowable gap, in milliseconds, between the current and previous data packets. If **DataTCNegTolerance** and **DataTCPosTolerance** values in milliseconds are set to **3000** and **-500** respectively, then the timecode difference between the currently arriving packet and the previous data packet must fall within 3000 and -500 milliseconds.

AVSyncToleranceEnable: Boolean value that monitors timecode jumps when set to **true**.

AVSyncTolerance: Timecode difference between the currently arriving packet and the previous data packet.

DebugLog: Boolean. When set to **true**, this turns on verbose logging for debugging.

MediaCaster/Properties

Defined MediaCaster properties for an application.

MediaReader/Properties

Defined MediaReader properties for an application.

MediaWriter/Properties

Defined MediaWriter properties for an application.

LiveStreamPacketizer/Properties

Defined LiveStreamPacketizer properties for an application.

HTTPStreamer/Properties

Defined HTTPStreamer properties for an application.

Manager/Properties

These properties are name/value pairs, which are supplemental properties used by Wowza Streaming Engine Manager. This element is included for future use.

Repeater/[OriginURL, QueryString]

Origin URL and query string to use when using the application as a live stream repeater. If the application is a DVR repeater, the **OriginURL** is the URL of the DVR origin server from which to retrieve audio and video chunks.

StreamRecorder/Properties

Supplemental configuration settings for tuning **StreamRecorder** instances. For more information, see the help text that accompanies property settings in Wowza Streaming Engine Manager.

Modules/Module/[Name, Description, Class]

List of modules available to this application. The **Modules** list must contain a unique **Name** element in order to identify the module. The **Description** field isn't used. The **Class** field is the full package name and class name of the module. For more information, see About server-side modules.

Properties

Typed name/value pairs. All application properties are copied to child application instances during instance creation. These properties are available in the Java API through the **IApplicationInstance.getProperties()** interface.