Wowza Media Server® 3

Configuration Reference

Wowza Media Server 3: Configuration Reference

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

Third Party Information

This document contains links to third party websites that are not under the control of Wowza Media Systems, LLC ("Wowza") and Wowza is not responsible for the content on any linked site. If you access a third party website mentioned in this document, then you do so at your own risk. Wowza provides these links only as a convenience, and the inclusion of any link does not imply that Wowza endorses or accepts any responsibility for the content on third party sites.
This document also refers to other third party software that is not licensed, sold, distributed or otherwise endorsed by Wowza. Please ensure that any and all use of Wowza® software and third party software is properly licensed.

Trademarks

Wowza, Wowza Media Systems, Wowza Media Server and related logos are either registered trademarks or trademarks of Wowza Media System, LLC in the United States and/or other countries.

Adobe and Flash are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Microsoft and Silverlight are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

QuickTime, iPhone, iPad and iPod are either registered trademarks or trademarks of Apple, Inc. in the United States and/or other countries.

Other product names, logos, designs, titles, words or phrases mentioned may be third party registered trademarks or trademarks in the United States and/or other countries.

Third party trademarks are used solely to identify and describe third party products as being compatible with Wowza products. Wowza is in no way sponsored, endorsed by or otherwise affiliated with any such third party trademark owners.

Third Party Copyright Notices

Log4j and Mina: Copyright © 2006, The Apache Software Foundation

Java Service Wrapper: Copyright © 1999, 2006, Tanuki Software, Inc.

Silver Egg Technology: Copyright © 2001, Silver Egg Technology

Java ID3 Tag Library and JLayer 1.0 (classic): Copyright © 1991, 1999, Free Software Foundation, Inc.

Bouncy Castle Crypto API: Copyright © 2000 – 2008, The Legion Of The Bouncy Castle

Apache Commons Lang libraries and Modeler libraries: Copyright © 2001-2008, The Apache Software Foundation

WebM VP8 Codec libraries: Copyright © 2010, Google Inc. All rights reserved.

Vorbis/Ogg libraries: Copyright © 2011, Xiph.org Foundation

Libgcc s-4 library and Libstdc++ library: Copyright © 2011, Free Software Foundation, Inc.

Speex Codec: Copyright © 2002-2003, Jean-Marc Valin/Xiph.org Foundation

Table of Contents

Introduction

This document is a reference for the configuration files that are located in the **[install-dir]/conf** folder of Wowza Media Server 3. It covers the most commonly used settings and configuration items. All internal configuration settings will be omitted.

Server Configuration File

Server.xml

Server level configuration.

CommandInterface/HostPort

HostPort definition for the Wowza Server command interface. This interface is currently only used to allow remote shutdown of the server. The entire **CommandInterface** section should be commented out when a server is deployed into production.

AdminInterface/ObjectList

List of objects made available through Java Management Extensions (JMX) interface. See the **Server Management Console and Monitoring** chapter of the **Wowza Media Server 3 User's Guide** for more information.

JMXRemoteConfiguration/[*]

Java Management Extensions (JMX) interface configuration. See the **Server Management Console and Monitoring** chapter of the **Wowza Media Server 3 User's Guide** for more information.

HandlerThreadPool/PoolSize, TransportThreadPool/PoolSize

Defines the maximum size of the server level threads in the transport and handler thread pools. The transport thread pool is used to read/write data from the transport sockets. The handler thread pool is used to process incoming messages. The Server level thread pools are only used if a virtual host's thread pool size is set to 0. This server level thread pool is also used to process the shutdown command. For this reason 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 and this value incrementing by 1. The most common value for RTSP/RTP based servers is 6970. If you plan on supporting RTSP/RTP, native RTP or MPEG-TS streams it is best to open up UDP ports 6970-9999.

RTP/DatagramPortSharing

If set to true then UDP ports (both unicast and multicast) can be shared between SDP files and MPEG-TS streams or shared between application instances. This 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 each referring to the unique audio stream but sharing the single RTP video 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 are not allowed to be shared and an error will be generated if an attempt is made to reuse a UDP port.

Properties

Properties are typed name/value pairs. These properties are available in the server side API through the **IServer.getProperties()** interface.

VHosts.xml

The **VHosts.xml** configuration file is used to define virtual host environments. By default the server ships with a single virtual host environment named **_defVHost_**. A complete description of this configuration file can be found in the **Virtual Hosting** chapter of the **Wowza Media Server 3 User's Guide**.

log4j.properties

The **log4j.properties** file is used to configure server logging. The server uses the Java based log4j logging system. By default the server is configured to log basic information to the console window and detail information in W3C Extended Common Log Format (ECLF) to log files. Detailed information on how to configure the logging system can be found in the **Logging** section of the **Server Administration** chapter of the **Wowza Media Server 3 User's Guide**.

Virtual Host Configuration Files

Authentication.xml

Authentication configuration for RTSP and HTTP sessions.

Method/[Name, Description, Class, Properties]

Definition for an authentication method. **Name** is the name of the method. **Class** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

DVR.xml

Definitions for Wowza nDVR recording and playback.

DVRRecorder/[Name, Description, BaseClass, Properties]

Definition for DVRRecorder item. **Name** is the name of the Recorder. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

DVRStore/[Name, Description, Properties]

Definition for DVRStore item. **Name** is the name of the Recorder. **Properties** are additional configuration properties.

HTTPStreamers.xml

Definitions for HTTP streaming such as Cupertino streaming, San Jose streaming and Smooth Streaming.

HTTPStreamer/[Name, Description, BaseClass, Properties]

Definition for HTTPStreamer item. **Name** is the name of the HTTPStreamer. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

HTTPStreamer/ApplicationContextClass

Full path to the class that is used to provide application instance context.

HTTPStreamer/RequestFilters

RequestFilters are used to determine if a given HTTP request should be processed by this HTTPStreamer.

HTTPStreamer/IdleFrequency
Time in milliseconds between idle events.

HTTPStreamer/LiveStreamPacketizer
Packetizer used by this HTTPStreamer.

HTTPStreamer/LiveStreamRepeater
Repeater used by this HTTPStreamer.

HTTPStreamer/Properties
Extra configuration properties for this HTTPStreamer.

LiveStreamPacketizers.xml
Definition for HTTP live stream packetizers that are used for HTTP streaming technology such as Cupertino streaming and Smooth Streaming.

LiveStreamPacketizer/[Name, Description, BaseClass, Properties]
Definition for LiveStreamPacketizer item. **Name** is the name of the LiveStreamPacketizer. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

LiveStreamPacketizer/IsRepeater
Set to **true** if this packetizer is a live stream repeater packetizer.

LiveStreamPacketizer/LiveRepeaterPlayerClass
Full path to class used on player side for live stream repeater.

LiveStreamPacketizer/LiveRepeaterReceiverClass
Full path to class used on receiver side for live stream repeater.

MediaCasters.xml
MediaCaster definitions.

Connections/[*]
Socket configuration for MediaCaster connections.

MediaCaster/[Name, Description, BaseClass, Properties]

Definition for MediaCaster item. **Name** is the name of the MediaCaster. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

MediaCaster/StreamType

Stream type in **Streams.xml** that is used when using this MediaCaster.

MediaCaster/ConnectionTimeout

Time in milliseconds this MediaCaster will wait when connecting to remote service.

MediaCaster/KeepAliveTime

Time in milliseconds this MediaCaster will remain running after the last user of the MediaCaster drops off before the MediaCaster is shutdown.

MediaReaders.xml

File format reader definitions.

MediaReader/[Name, Description, BaseClass, Properties]

Definition for MediaReader item. **Name** is the name of the MediaReader. The **Name** doubles as the stream name prefix that is used to invoke this MediaReader. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

MediaReader/FileExtension

Default file extension.

MediaWriters.xml

File format writer definitions.

MediaWriter/[Name, Description, BaseClass, Properties]

Definition for MediaWriter item. **Name** is the name of the MediaWriter. The **Name** doubles as the stream name prefix that is used to invoke this MediaWriter. **BaseClass** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

MP3Tags.xml

Map for mapping MP3 id3 tags to metadata property names.

MP3Tag/[ID, Name]

Maps internal MP3 ID3 tag ids (**ID**) to metadata property names (**Name**).

RTP.xml

List of de-packetizers for handling native RTP and MPEG-TS incoming streams.

DePacketizer /[Name, Description, Class, Properties]

Definition for DePacketizer item. **Name** is the name of the DePacketizer. **Class** is the full path to the class that provides the underlying implementation. **Properties** are additional configuration properties.

StartupStreams.xml

List of streams to startup and virtual host startup.

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_** will be used.

StartupStream/MediaCasterType

MediaCaster type name from **MediaCasters.xml** to use to start the stream.

StartupStream/ StreamName

Name of stream to startup. Should include stream name prefix if needed.

Streams.xml

Definition of stream types.

Stream/[Name, Description, ClassBase, ClassPlay, Properties]

Definition for Stream item. **Name** is the name of the Stream. **BaseClass** is the full path to the class that provides the underlying implementation for the IMediaStream interface. **PlayClass** is the full path to the class that provides the underlying implementation for the IMediaStreamPlay interface. **Properties** are additional configuration properties.

VHost.xml

Virtual host configuration.

HostPortList/HostPort

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

HostPortList/HostPort/ProcessorCount

Number of threads to use to service connections. See the online **General Tuning Guide** for recommended values based on server resources.

HostPortList/HostPort/[IpAddress, Port]

IpAddress is the IP address or domain name of the address to which you want to listen to for incoming requests. If **IpAddress** is set to the wildcard character (*****) Wowza Server will attempt to listen for incoming connections on all available network interfaces. **Port** is a comma separated list of ports.

HostPortList/HostPort/SocketConfiguration/[*]

This section is the detailed configuration of the socket connection that is created by this **HostPort** definition at runtime. It is through these settings that you can tune the performance of the socket connections that will be used to send data into and out of Wowza Server. The **SendBufferSize** and **ReceiveBufferSize** are the two most important settings in this group. They define the size of the memory buffers used during data transfer over the socket connection. See the online **General Tuning Guide** for recommended values based on server resources.

The **ReuseAddress** and **KeepAlive** settings should both be set to true and 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. The Wowza Server server will respond to TCP connection requests as quickly as possible. This value should not be set to a value less than 50. It can be set to a value of -1 which will allow the operating system to control the value (this is not always the best setting, some platforms will then use a very small value for this which may 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, smoothstreaming.

HostPortList/HostPort/HTTPProviders

List of HTTPProviders that this **HostPort** will include when processing HTTP requests. See the **HTTPProviders** section of the **Server-side Modules and Extensions** chapter of the **Wowza Media Server 3 User's Guide** for more information.

HostPortList/HostPort/SSLConfig/[*]

SSL configuration for a given **HostPort**. **KeyStorePath** is the full path to the key-store file. **KeyStorePassword** is the key-store password. **KeyStoreType** is the key-store type (default value is JKS for Sun Java JRE). **Algorithm** is the encryption algorithm (default value is SunX509 for Sun Java JRE).

HandlerThreadPool/PoolSize, TransportThreadPool/PoolSize

Defines the maximum size of the virtual host level threads in the transport and handler thread pools. The transport thread pool is used to read/write data from the transport sockets. The handler thread pool is used to process incoming messages. If the pool size is set to zero for a given thread pool type, the server level thread pool of the same type will be used for this virtual host. See the online **General Tuning Guide** for recommended values based on server resources.

IdleWorkers/[WorkerCount, CheckFrequency]

IdleWorkers/WorkerCount controls the number of threads being used to generate idle events. **IdleWorkers/CheckFrequency** is the time in milliseconds between checking to see if a client has been idle for **Client/IdleFrequency**. The **IdleWorkers/CheckFrequency** should be at least four times smaller than the **Client/IdleFrequency**. See the online **General Tuning Guide** for recommended values based on server resources.

NetConnections/[ProcessorCount, IdleFrequency, SocketConfiguration]

Tuning of connections made between Wowza Server servers (for example when using the live stream repeater).

HTTPTunnel/KeepAliveTimeout

This is the keep alive time for RTMPT, RTMPTE and RTMPS connections.

Client/[ClientTimeout, IdleFrequency]

ClientTimeout is the time in milliseconds the server will wait before shutting down a non-responding client connection. **IdleFrequency** is the time in milliseconds between idle events. For basic video on demand streaming a value of 250 milliseconds will provide the best reliability versus performance ratio. For live streaming a value of between 125 and 250 milliseconds is more desirable. It will increase the frequency at which media data is sent to the Flash client. If you adjust this value, be sure to also adjust the **IdleWorkers/IdleFrequency** to a value that is at least four times smaller.

RTP/IdleFrequency

Time in milliseconds between idle events for RTSP connections.

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

Datagram socket configuration for incoming and outgoing RTP connections. **ReceiveBufferSize** is the size in bytes of the incoming UDP buffer. **SendBufferSize** size in bytes of the outgoing UDP buffer. **MulticastTimeout** is the timeout in milliseconds for multicast polling. **DatagramMaximumPacketSize** is maximum size in bytes for single incoming UDP packet.

RTP/[*]/ProcessorCount

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

Application/ApplicationTimeout

The time in milliseconds the server will wait before shutting down an application to which no clients are connected. A value of zero will keep applications running until the virtual host is shutdown.

Application/PingTimeout

The time in milliseconds the server will wait for a ping response from the client. The ping mechanism is an RTMP internal ping (not and ICMP ping) that is used to validate a client connection. If set to zero, the server will wait forever.

Application/ValidationFrequency

The time in milliseconds the server will wait between 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 set to zero, the server will stop validating client connections.

Application/MaximumPendingWriteBytes

Maximum number of bytes that are allowed to be queued up waiting to be sent. If this value is exceeded for an individual client connection then the connection is terminated. If set to zero there is no maximum value.Application/MaximumStorageDirDepth

Maximum allowed number of sub-folders allowed in any storage path. This is to protect against symbolic link loops.

Properties

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

Application Configuration Files

Application.xml

Application configuration.

ApplicationTimeout

The time in milliseconds the server will wait before shutting down an application to which no clients are connected. A value of zero will keep applications running until the virtual host is shutdown. If this value in not provided (section commented out) the value set in the **VHost.xml** will be used.

PingTimeout

The time in milliseconds the server will wait for a ping response from the client. The ping mechanism is an RTMP internal ping (not and ICMP ping) that is used to validate a client connection. If set to zero, the server will wait forever. If this value in not provided (section commented out) the value set in the **VHost.xml** will be used.

ValidationFrequency

The time in milliseconds the server will wait between 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 set to zero, the server will stop validating client connections. If this value in not provided (section commented out) the value set in the **VHost.xml** will be used.

MaximumPendingWriteBytes

Maximum number of bytes that are allowed to be queued up waiting to be sent. If this value is exceeded for an individual client connection then the connection is terminated. If set to zero there is no maximum value. If this value in not provided (section commented out) the value is set in the **VHost.xml** will be used.

MaximumSetBufferTime

The maximum number of milliseconds honored server side for client side call to NetStream.setBufferTime(secs). Set this value to zero to turn off this check. The default value is 60000 (or 60 seconds). This is to combat Replay Media Catcher which will set a very large client side buffer to trick the 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 allow number of sub-folders allowed in any storage path. This is to protect against symbolic link loops. If this value in not provided (section commented out) the value set in the **VHost.xml** will be used.

Connections/AutoAccept

This setting determines if the application will automatically accept incoming Adobe Flash player connection request. If **true** all incoming connection request will automatically be accepted. If **false** the application is required to make a server side call to **client. acceptConnection()** to accept an incoming connection request.

Connections/AllowDomains

Comma delimited list of domain names or IP address for which client connections will be accepted. The domain names or IP addresses that are specified here represent the domain name or IP address of the Adobe Flash swf file that is connecting to the Wowza Server server or the IP address of the client connecting to Wowza Server. If this value is left empty then connections from all domains or IP addresses are accepted. For example if you have a .swf file that is located at the url:

http://www.mycompany.com/flash/myflashmovie.swf

To configure your server such that only content from your domain can access your Wowza Server server you would set AllowDomains to **www.mycompany.com**. 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 are working with a client side encoder such as On2 Flix Live which does not provide a valid referrer.

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

The allow domains processing occurs just before the event method onConnect. So if you would like to provide more fine 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

The name (as defined in the **Streams.xml** file) of the default stream type for this application. An explanation of stream types can be found in the **Stream Types** section of **Application Configuration** chapter in the User's Guide.

Streams/StorageDir, Streams/KeyDir, SharedObjects/StorageDir

Streams/StorageDir is the full path to the directory where the application will read and write media files. **Streams/KeyDir** is the directory where the Cupertino streaming AES-128 encryption keys are stored. **SharedObjects/StorageDir** is the full path to the directory where the application will read and write remote stored object data. If these values are left blank, an application will use 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:

```
${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 any incoming live streams. Live stream packetization is done to make a stream available for HTTP streaming to the Apple iPhone and iPod touch, Adobe Flash and Microsoft Silverlight. Live Stream Packetization is also done to DVR a live stream. **LiveStreamPacketizers** can contain none, one or more of the following values (separated by commas):

LiveStreamPacketizers	Description
cupertinostreamingpacketizer	Cupertino: iOS devices
smoothstreamingpacketizer	Smooth: Microsoft Silverlight
sanjosestreamingpacketizer	San Jose: Adobe Flash
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
dvrstreamingpacketizer	Wowza nDVR: Streaming
dvrstreamingrepeater	Wowza nDVR: Live stream repeater

Streams/Properties

Properties defined here will override properties defined in **Streams.xml** for this application.

Transcoder/LiveStreamTranscoder

The name (as defined in the **LiveStreamTranscoders.xml** file) of the default Transcoder handle for this application. If set to **transcoder**, the live stream transcoder will be enabled for this application. If left blank, incoming streams will not be transcoded.

Transcoder/Templates

Transcoder/Templates is the list of template names to search when matching an incoming stream to a live stream transcoder template. If the live stream transcoder is enabled, each new live stream that is published to the application will be a candidate to be transcoded. The Wowza Server will search the **Transcoder/Templates** for the first template file that exists. Once it finds an existing transcoder template, it will use that template to transcode the stream. If no matches are found, the stream will not be transcoded. The default value for this setting is:

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

The first item in the list uses the variable **${SourceStreamName}**. If there is a transcoder template in the template directory with the name **[stream-name].xml** (where **[stream-name]** is the name of the incoming stream), then that transcoder template will be used. If this file does not exists, the **transrate.xml** template will be used if it exists. If neither exists, then the stream will not be transcoded. Only the variable **${SourceStreamName}** is supported.

Transcoder/ProfileDir

Transcoder/ProfileDir is not currently used and is for future use:

Transcoder/TemplateDir

Transcoder/TemplateDir is the full path to the directory where the application will look for transcoder templates used to control the live stream transcoder. By default Wowza Server will look for transcoder templates in the **[install-dir]/transcoder/templates** folder. You can also define a folder per-application by using path variables. For example, if you would like to setup a per-application templates folder, specify the path:

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

The following variables are supported:

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

DVR/Recorders

The name (as defined in the **DVR.xml** file) of the default DVR recorder for this application.

DVR/Store

The name (as defined in the **DVR.xml** file) of the default DVR store for this application.

DVR/WindowDuration

The size of material available from Wowza nDVR in seconds. A **WindowDuration** of 0 (default) denotes that there is no window duration and all DVR-ed data is available.

DVR/StorageDir

DVR/StorageDir is the full path to the directory where the application will read and write Wowza nDVR data.

The following variables are supported:

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

DVR/ArchiveStrategy

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

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

DVR/Repeater/ChunkOriginURL

The URL of the DVR origin server to retrieve audio and video chunks. This is used when the Application is a DVR repeater.

DVR/Properties

Properties defined here will override properties defined in **DVR.xml** for this application.

HTTPStreamers

Available HTTP streaming types. HTTP is done to make a stream available for HTTP streaming to the Apple iPhone and iPod touch, Adobe Flash and Microsoft Silverlight. It also allows the 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):

LiveStreamPacketizers	Description
cupertinostreaming	Cupertino: HTTP streaming to iPhone and iPod touch
smoothstreaming	Smooth: HTTP streaming to Microsoft Silverlight
sanjosestreaming	San Jose: HTTP streaming to Adobe Flash
dvrchunkstreaming	DVR: Enable streaming from origin to edge

Client/IdleFrequency

Time in milliseconds between idle events. Idle events represent the heart-beat of streaming for Adobe Flash and RTSP/RTP streaming. It presents how often new data is sent to the player. If this value is set to -1 then the value specified in **VHost.xml** will be used.

Client/Access/[*]

Controls the default access an Adobe Flash client connection has to assets connected to a particular Wowza Server application. An individual client's access can be modified through the server side API. This is most commonly done in the **onConnect** or **onConnectAccept** event handler. Each of these settings is a comma delimited list of names that are matched against the asset name (stream name or shared object name) to control access. If any part of the asset name matches one of the elements in the list match then the given access is granted. The values are case sensitive. If the parameter is empty (blank) then access is denied to all clients. If the parameter is set to the (*****) 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:

```
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: controls access to view or listen to a NetStream object.

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

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

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

SharedObjectReadAccess: controls access to read values from a RemoteSharedObject.

SharedObjectWriteAccess: controls access to write values to a RemoteSharedObject.

RTP/Authentication/[PublishMethod, PlayMethod]

The authentication method used to secure RTSP connections to Wowza Server. **PublishMethod** is for incoming or publishing connections and **PlayMethod** is for outgoing or play connections. Authentication methods are defined and configured in **Authentication.xml**. By default there are three authentication methods: **none** (no authentication), **basic** (password and username are sent in clear text) and **digest** (password is hashed using MD5 and is never sent in clear text over the network). Usernames and passwords are stored in the file **conf/publish.password**. The format of this file is a line per user with the username first followed by a space followed by the password. The authentication method can also be set at the virtual host level in **VHost.xml**.

RTP/[AVSyncMethod, MaxRTCPWaitTime]

Control how Wowza Server synchronizes the audio and video channels when receiving a RTP stream. **AVSyncMethod** configures the methodology used to synchronize the audio and video channels. There are three possible values; **senderreport** (use the Sender Report (SR) packets that

are sent over the Real-time Control Protocol (RTCP) channel), **rtptimecode** (assume the RTP timecodes are absolute timecode values), **systemclock** (synchronize based on the system clock). The default value is **senderreport**. **MaxRTCPWaitTime** is the maximum time in milliseconds Wowza Server will wait to receive a Sender Report (SR) packet over the Real-time Control Protocol (RTCP) channel. If no SR packets are received within this time period the server will default to using the **rtptimecode** method.

RTP/IdleFrequency

The time in milliseconds between RTP idle events. Idle events are used to send new media data and events to an RTP or MPEG-TS session.

RTP/RTSPSessionTimeout

The time in milliseconds by which an idle RTSP session will be determined to be stale and will be disconnected. To turn off idle disconnect, set this value to zero. Wowza will monitor all RTSP sessions looking for periodic RTSP messages or RTCP receiver packets over RTP. If the server has not received messages during the session timeout period a session will be disconnected.

RTP/RTSPMaximumPendingWriteBytes

The maximum number of bytes that can be pending waiting to be written to an RTSP session. If an RTSP session has more bytes waiting to be written then what is specified by this setting the session will be disconnected. Wowza Server 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 value specified then the session is determined to be dead and the session is disconnected. To turn off maximum pending write byte monitoring, set this value to zero.

RTP/[RTSPBindIpAddress, RTSPConnectionIPAddress, RTSPOriginIpAddress]

Control the IP addresses exchanged and used during RTSP/RTP port and IP address negotiation when the RTP portion of the stream is being delivered over UDP. The **RTSPBindIpAddress** property is the IP address that Wowza Server binds to when delivering RTP packets over UDP. If the server is using network address translation routing (NAT) then this address should be set to the internal IP address of the network interface that is mapped to the external IP address. If NAT is not being used, then this value should be set to the external IP address of the server. The **RTSPConnectionIpAddress** and **RTSPOriginIpAddress** are the two 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/Properties

Properties defined here will override properties defined in **RTP.xml** for this application.

MediaCaster/Properties

Properties defined here will override properties defined in **MediaCasters.xml** for this application.

MediaReader/Properties

Properties defined here will override properties defined in **MediaReaders.xml** for this application.

MediaWriter/Properties

Properties defined here will override properties defined in **MediaWriters.xml** for this application.

LiveStreamPacketizer/Properties

Properties defined here will override properties defined in **LiveStreamPacketizers.xml** for this application.

HTTPStreamer/Properties

Properties defined here will override properties defined in **HTTPStreamers.xml** for this application.

Repeater/[OriginURL, QueryString]

Origin URL and query string to use when using the live stream repeater.

Modules/Module/[Name, Description, Class]

List of modules that are available to this application. The **Name** element must be unique within the **Modules** list and is used to identify the module. The **Description** field it not used. The **Class** field is the full package name and class name of the module. Please see the **Server-side Modules and Extensions** chapter of the **Wowza Media Server 3 User's Guide** for more information.

Properties

Properties are typed name/value pairs. All application properties are copied to child application instances upon instance creation. These properties are available in the server side API through the **IApplicationInstance.getProperties()** interface.