Wowza Media Server® Pro

Wowza Pro Media Security

Wowza Media Server Pro: Media Security

Third-Party Information

This document contains links to third-party websites that are not under the control of Wowza Media Systems, Inc. ("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.

Trademarks

Wowza, Wowza Media Systems, Wowza Media Server and related logos are trademarks of Wowza Media Systems, Inc., and may be registered in the United States or in other jurisdictions including internationally.

Adobe and Flash are registered trademarks of Adobe Systems Incorporated, and may be registered in the United States or in other jurisdictions including internationally.

Other product names, logos, designs, titles, words, or phrases mentioned may be trademarks, service marks or trade names of other entities and may be registered in certain jurisdictions including internationally.

Third Party Copyright Notices

Log4j and Mina: Copyright © 2006 The Apache Software Foundation

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

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

Table of Contents

Introduction

This document covers the available options and procedures for securing Wowza Media Server Pro and the media you plan to stream through Wowza Pro. There are several Wowza Pro features that are needed to properly secure your content. Some of these are SecureToken, SecureURLParams, RTSP Authentication and StreamNameAlias. The methods required to secure Wowza Pro differ based on the method used to stream media. For example to protected video on demand streaming it is best to use SecureToken while protecting live streaming may require a combination of SecureToken, SecureURLParams, RTSP Authentication and StreamNameAlias. The SecureToken and SecureURLParams security features are included in this package. These two features can be used in tandem and are installed and configured together. RTSP Authentication is built into Wowza Pro. The StreamNameAlias add on package is a separate package that can downloaded from the following forum thread:

http://www.wowzamedia.com/forums/showthread.php?t=1505

Note

This security package requires Wowza Media Server Pro 1.6.0 patch23 or greater.

Overview

Below is a description of each of the package covered in this document.

SecureToken

SecureToken is a challenge and response based security system that provides a high level of content protection against spoofing threats like those posed by the "Replay Media Catcher". Each connection is protected by a random single use key and a password (shared secret). The basic security methodology is described below.

The way SecureToken works is that upon client connection the provided custom module generates a unique key for the pending connection. The generated key is encrypted using the TEA (Tiny Encryption Algorithm) algorithm using a password (shared secret) that is shared between the Wowza Pro server and your Flash client movie. The encrypted unique key is returned as the "secureToken" parameter of the object that is the first parameter to the event callback NetConnection.onStatus. The Flash client movie then decrypts the unique key using the shared password and sends the result back to the custom module by calling NetConnection.call("secureTokenResponse", null, decodedKey). The server then compares this key to the originally generated key. If they match then processing for that connection continues. If the values do not match then the connection is aborted. If the Flash client movie tries to create

a NetStream object without first calling "secureTokenResponse" with the correctly decoded key, then the connection is aborted.

SecureURLParams

SecureURLParams is a simple method for providing password like protection to the connection process between the Flash player client and Wowza Pro. This security measure is generally used to protect the publishing process. In most cases it is used along side SecureToken. SecureURLParams is used to protect publishing and SecureToken is used to protect playback.

With SecureURLParams you define name value pair combinations that are assigned to protect one of three server functions: connect, play or publish. The name value pairs are passed to Wowza Pro as part of the rtmp connection url (works with all variants of rtmp such as rtmpe).

StreamNameAlias

StreamNameAlias is system for creating simple aliases that can expand into more complicated steram names. This package can also be used to limit the stream names that can be accepted by Wowza Pro. This is the feature of this package that we will be leveraging to help protect live re-streaming. The complete details of how this system works is covered in detail in the StreamNameAlias documentation that accompanies the package.

RTSP Authentication

RTSP Authentication provides a means for password protecting the RTSP/RTP publishing interface into Wowza Pro. This feature is built into Wowza Pro. Authentication is configured on a per-application basis using the RTP/Authentication/Method property in Application.xml. Usernames and passwords are stored in the "[install-dir]/conf/rtp.password" file. This feature is covered in more detail in the Wowza Media Server Pro User's Guide.

Installation

To install SecureToken and SecureURLParams copy the file "lib/wms-plugin-security.jar" from this package to the "[install-dir]/lib" folder of the Wowza Pro server. SecureToken and SecureURLParams configuration is done on a per-application basis.

Configuration

Server Side Configuration

Server side SecureToken and SecureURLParams are configured together. They are both configured on a per-application basis in an application's "[install-dir]/conf/[application]/Application.xml" file (where [application] is the name of the application

you wish to protect). Below are the steps to create a new application named "securestreaming" and add the "ModuleSecureURLParams" module:

1. Create the folder "[install-dir]/applications/securestreaming"

2. Create the folder "[install-dir]/conf/securestreaming" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file and add the following <Module> definition as the last entry in the <Modules> list:

```
<Module>
     <Name>ModuleSecureURLParams</Name>
     <Description>ModuleSecureURLParams</Description>
     <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
```

With this in place the "securestreaming" application is completely locked down. Nobody can connect to it. Next, we will examine how to secure entry points into Wowza Pro.

There are basically three different entry points into Wowza Pro that are secured using SecureToken and SecureURLParams. They are "connect", "play" and "publish". SecureToken can be used to protect the "play" and "publish" entry points. SecureURLParams can be configured to protect all three entry points. If a SecureToken password (shared secret) is configured then both "play" and "publish" are protected by SecureToken unless a SecureURLParam is defined for that entry point. The SecureToken password and the SecureURLParams are configured by adding properties to the application level <Properties> defined at the bottom of the Application.xml file (be careful there are more than one properties containers in Application.xml):

secureTokenSharedSecret

This is the password (shared secret) that is used to encrypt/decrypt the token used to secure the connection. If this property is set in "Application.xml", then SecureToken will be used to protect both the "play" and "publish" entry points unless a SecureURLParam is defined for that entry point. For example:

```
<Properties>
  <Property>
     <Name>secureTokenSharedSecret</Name>
     <Value>#ed%h0#w@1</Value>
  </Property>
</Properties>
```

secureurlparams.connect

This is a comma separated list of security parameters in the form [value].[key] used to protect the "connect" entry point into the Wowza Pro server.

```
<Properties>
   <Property>
      <Name>secureurlparams.connect</Name>
      <Value>1gt345.doConnect</Value>
   </Property>
</Properties>
```

secureurlparams.play

This is a comma separated list of security parameters in the form [value].[key] used to protect the "play" entry point into the Wowza Pro server.

```
<Properties>
   <Property>
      <Name>secureurlparams.play</Name>
      <Value>1gt345.doPlay</Value>
   </Property>
</Properties>
```

secureurlparams.publish

This is a comma separated list of security parameters in the form [value].[key] used to protect the "publish" entry point into the Wowza Pro server.

```
<Properties>
   <Property>
      <Name>secureurlparams.publish</Name>
      <Value>1gt345.doPublish</Value>
   </Property>
</Properties>
```

The next section illustrates how to code your Flash player to handle SecureToken and SecureURLParams.

Client Side Configuration

Client side configuration and coding is a little less straight forward. The means for passing SecureURLParams and responding to the SecureToken request are going to differ based on your Flash player code, pre-built Flash player or the encoder you are using. In this section we will cover the basics and highlight some of the more common scenarios.

Basic SecureToken Example

So let's start with the SecureToken basics. For this example we will setup an Application.xml that uses the "ModuleSecureURLParams" module and sets the "secureTokenSharedSecret" to "mytestpassword". The <Properties> at the bottom of Application.xml will look like this:

```
<Properties>
   <Property>
      <Name>secureTokenSharedSecret</Name>
      <Value>mytestpassword</Value>
   </Property>
</Properties>
```

The Flash player code to make a secure connection to the server looks like this:

```
import com.meychi.ascryptAS3.TEA;

var nc:NetConnection = new NetConnection();
function ncOnStatus(infoObject:NetStatusEvent)
{
   if (infoObject.info.code == "NetConnection.Connect.Success")
   {
      if (infoObject.info.secureToken != null)
         nc.call("secureTokenResponse", null,
            TEA.decrypt(infoObject.info.secureToken, "mytestpassword"));
   }
}
nc.addEventListener(NetStatusEvent.NET_STATUS, ncOnStatus);
nc.connect("rtmp://localhost/securestreaming");
```

The first line of this example imports the TEA library that is used to decrypt the SecureToken token. ActionScript 2 and 3 versions of this code are included in the "client/com" folder of this package. If you plan on integrating this code into your player, you will need to copy these classes into your Flash code. Next, we define and create a NetConnection object that will be used to communicate with Wowza Pro. The next function is the NetConnection onStatus handler that will be invoked during the lifecycle of the NetConnection object. Next, we add onStatus handler as a listener to the NetConnection object and finally all NetConnection.connect(url) to connect to the Wowza Pro server.

When the NetConnection object establishes a connection with Wowza Pro, the onStatus handler will be called with a infoObject.info.code value of "NetConnection.Connection.Success". If the server is protected with SecureToken then the infoObject.info object will also contain a SecureToken challenge in the "secureToken" field. To respond to this challenge, the Flash player code calls the remote function "secureTokenResponse" with the first parameter set to the decrypted token. You can see we are decrypting the token with the call:

```
TEA.decrypt(infoObject.info.secureToken, "mytestpassword")
```

This is all that needs to be done to complete the challenge and response cycle. Once the call is made to "secureTokenResponse" the connection is validated and the rest of your Flash code will execute normally. If the "secureTokenResponse" function is not called before you Flash player code calls "play" or "publish" then Wowza Pro will terminate the connection.

If you are using a pre-built player such as JW Player or FlowPlayer or one of the Flash playback components such as the FLVPlayback component then this code will need to be inserted. The "client" folder of this package includes examples of how to integrate this code into the FLVPlayback component and the FastPlayVideoStreaming example player. JW Player comes with SecureToken built in. This forum post describes how to use this built in feature:

http://www.wowzamedia.com/forums/showthread.php?t=1665

Basic SecureURLParams Example

Now let's tackle SecureURLParams. For this example let's define two SecureURLParams one to protect "connect" "12345.doConnect" and one to protect "publish" "54321.doPublish". The <Properties> at the bottom of Application.xml will look like this:

```
<Properties>
   <Property>
      <Name>secureurlparams.connect</Name>
      <Value>12345.doConnect</Value>
   </Property>
   <Property>
      <Name>secureurlparams.publish</Name>
      <Value>54321.doPublish</Value>
   </Property>
</Properties>
```

The Flash player code to make a secure connection to the server for publishing looks like this:

```
var nc:NetConnection = new NetConnection();
nc.connect("rtmp://localhost/securestreaming?doConnect=12345&doPublish=54321");
```

You can see that the SecureURLParams are passed to Wowza Pro as query parameters tacked on to the connection url. The Flash player will send these query parameters to Wowza Pro as part of the connection process. Wowza Pro will validate these query parameters at each entry point. For example, when the internal "connect" method is called if a "connect" query parameter is configured in Application.xml and it matches the query parameter data then the connection will be processed normally. If they do not match then the connection will be terminated.

Note

Some live media encoders do not allow query parameters as part of the connection url (Flash Media Live Encoder (FMLE) being one example). If this is the case, you can add the query parameters as separate path entries appended to the end of the connection url and Wowza Pro will convert these to query parameters. If you use this method you will need to specify the full connection url including the application instance name. The default application instance name is _definst_. For example the url:

rtmp://[server-ipaddress]/streamtest?doConnect=12345&doPublish=54321

changes to:

rtmp://[server-ipaddress]/streamtest/_definst_/doConnect=12345/doPublish=54321.

Example Configurations

Below are some example configurations that cover how to combine SecureToken, SecureURLParams, RTSP Authentication and StreamNameAlias to protect different forms of streaming.

Video On Demand

This example illustrates how to protect video on demand streaming. Basic configuration:

```
Stream type: default
Add On Packages: SecureToken
```

Only SecureToken is needed to protect video on demand streaming. The steps are to create an application named "vod" are:

1. Create the folder "[install-dir]/applications/vod"

2. Create the folder "[install-dir]/conf/vod" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file and add the following <Module> definition as the last entry in the <Modules> list:

```
<Module>
     <Name>ModuleSecureURLParams</Name>
     <Description>ModuleSecureURLParams</Description>
     <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
```

4. In the same Application.xml add the following property to the application level properties at the bottom of the file:

```
<Properties>
  <Property>
     <Name>secureTokenSharedSecret</Name>
     <Value>#ed%h0#w@1</Value>
  </Property>
</Properties>
```

You will need to integrate the client side SecureToken response into your Flash player code.

Live Streaming using RTMP Encoder

This example illustrates how to protect live streaming when using an RTMP based encoder such as Flash Media Live Encoder or On2 Flix Live. Basic configuration:

```
Stream type: live
Add On Packages: SecureToken, SecureURLParams
RTP/Authentication/Method: digest
```

```
rtp.password file: empty
```

SecureToken and SecureURLParams are needed to protect live streaming. The steps are to create an application named "live" are:

1. Create the folder "[install-dir]/applications/live"

2. Create the folder "[install-dir]/conf/live" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file, change the Streams/StreamType to "live", verify the RTP/Authentication/Method is set to "digest" and add the following <Module> definition as the last entry in the <Modules> list:

```
<Module>
     <Name>ModuleSecureURLParams</Name>
     <Description>ModuleSecureURLParams</Description>
     <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
```

4. In the same Application.xml add the following properties to the application level properties at the bottom of the file:

```
<Properties>
  <Property>
    <Name>secureTokenSharedSecret</Name>
    <Value>#ed%h0#w@1</Value>
  </Property>
  <Property>
    <Name>secureurlparams.publish</Name>
    <Value>12345.doPublish</Value>
  </Property>
</Properties>
```

You will need to integrate the client side SecureToken response into your Flash player code. You will also need to use the "doPublish=12345" query parameter when connecting using the encoder. See note above about how to properly specify query parameters in Flash Media Live Encoder. For example to connect using Flash Media Live Encoder the "FMS URL" is:

rtmp://[wowza-ip-address]/live/_definst_/doPublish=12345

Live Streaming using RTSP/RTP Encoder

This example illustrates how to protect live streaming when using an RTSP/RTP based encoder such as Telestream Wirecast or Quicktime Broadcaster. Basic configuration:

```
Stream type: live
Add On Packages: SecureToken
RTP/Authentication/Method: digest
rtp.password file: add username/password entries for publishing
```

SecureToken is needed to protect this form of live streaming. The steps are to create an application named "live" are:

1. Create the folder "[install-dir]/applications/live"

2. Create the folder "[install-dir]/conf/live" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file, change the Streams/StreamType to "live" , verify the RTP/Authentication/Method is set to "digest" and add the following <Module> definition as the last entry in the <Modules> list:

```
<Module>
     <Name>ModuleSecureURLParams</Name>
     <Description>ModuleSecureURLParams</Description>
     <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
```

4. In the same Application.xml add the following property to the application level properties at the bottom of the file:

```
<Properties>
   <Property>
     <Name>secureTokenSharedSecret</Name>
     <Value>#ed%h0#w@1</Value>
   </Property>
</Properties>
```

You will need to integrate the client side SecureToken response into your Flash player code. Add username/password pairs to the "[install-dir]/conf/rtp.password" file for each user that is to be authorized to publish to this application. The username/password information will need to entered into the encoder streaming configuration parameters.

Live Streaming using Native RTP Encoder

This example illustrates how to protect live streaming when using a native RTP encoder that generates an SDP file. Basic configuration:

```
Stream type: rtp-live
Add On Packages: SecureToken, StreamNameAlias
RTP/Authentication/Method: digest
rtp.password file: empty
StreamNameAliasMap.txt: empty
```

SecureToken and StreamNameAlias are needed to protect this form of live streaming. You will need to download and install the StreamNameAliasPackage from here:

http://www.wowzamedia.com/forums/showthread.php?t=1505

The steps are to create an application named "rtplive" are:

1. Create the folder "[install-dir]/applications/rtplive"

2. Create the folder "[install-dir]/conf/rtplive" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file, change the Streams/StreamType to "rtp-live" , verify the RTP/Authentication/Method is set to "digest" and add the following <Module> definitions as the last entries in the <Modules> list (the order of these two modules is important):

```
<Module>
      <Name>ModuleSecureURLParams</Name>
      <Description>ModuleSecureURLParams</Description>
      <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
<Module>
   <Name>ModuleStreamNameAlias</Name>
   <Description>ModuleStreamNameAlias</Description>
   <Class>com.wowza.wms.plugin.streamnamealias.ModuleStreamNameAlias</Class>
</Module>
```

4. In the same Application.xml add the following property to the application level properties at the bottom of the file:

```
<Properties>

   <!-- ModuleStreamNameAlias Properties -->
   <Property>
      <Name>streamNameAliasFile</Name>
      <Value>${com.wowza.wms.context.VHostConfigHome}/conf/StreamNameAliasMap.txt</Value>
   </Property>
   <Property>
      <Name>streamNameAliasPathDelimiter</Name>
      <Value>/</Value>
   </Property>
   <Property>
      <Name>streamNameAliasNameDelimiter</Name>
      <Value>=</Value>
   </Property>
   <Property>
      <Name>streamNameAliasDebug</Name>
      <Value>true</Value>
      <Type>Boolean</Type>
   </Property>

   <!-- ModuleSecureURLParams Properties -->
   <Property>
      <Name>secureTokenSharedSecret</Name>
      <Value>#ed%h0#w@1</Value>
   </Property>

</Properties>
```

5. Edit "[install-dir]/conf/StreamNameAliasMap.txt" and empty out the file so there are no entries in the file.

You will need to integrate the client side SecureToken response into your Flash player code.

Live Streaming Origin/Edge

This example illustrates how to configure the edge servers in an origin/edge configuration to connect to an origin server that employs either SecureToken or SecureURLParams protection. All of the configuration covered in this example pertains to the edge server and not the origin server. First, be sure the SecureToken/SecureURLParams library "lib/wms-plugin-security.jar" has been copied into the "[install-dir]/lib" folder of each of the edge servers.

If the origin server is protected using SecureToken, you will need to add the following property to the application level properties at the bottom the Application.xml file of the edge application:

```
<Properties>
  <Property>
    <Name>secureTokenOriginSharedSecret</Name>
    <Value>#ed%h0#w@1</Value>
  </Property>
</Properties>
```

Set this property to the SecureToken password (shared secret) used to protect the origin.

If the edge is protected using SecureURLParams, you will need to set the Repeater/QueryString value in the edge application's Application.xml file to the query string needed to connect to the origin to play a stream. For example if "play" is protected on the origin by the query string "doConnect=12345&doPlay=54321", set the Repeater/Query to:

```
<QueryString><![CDATA[doConnect=12345&doPlay=54321]]></QueryString>
```

It is necessary to surround the query string value in a CDATA block because it contains the ampersand character which must be properly escaped when used in XML data.

SHOUTcast Re-Streaming

This example illustrates how to protect live streaming when re-streaming a SHOUTcast stream. Basic configuration:

```
Stream type: shoutcast
Add On Packages: SecureToken, StreamNameAlias
RTP/Authentication/Method: digest
rtp.password file: empty
StreamNameAliasMap.txt: entry for each SHOUTcast URL
```

SecureToken and StreamNameAlias are needed to protect this form of re-streaming. You will need to download and install the StreamNameAliasPackage from here:

http://www.wowzamedia.com/forums/showthread.php?t=1505

The steps are to create an application named "shoutcast" are:

1. Create the folder "[install-dir]/applications/shoutcast"
2. Create the folder "[install-dir]/conf/shoutcast" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file, change the Streams/StreamType to "shoutcast" , verify the RTP/Authentication/Method is set to "digest" and add the following <Module> definitions as the last entries in the <Modules> list (the order of these two modules is important):

```
<Module>
      <Name>ModuleSecureURLParams</Name>
      <Description>ModuleSecureURLParams</Description>
      <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
<Module>
   <Name>ModuleStreamNameAlias</Name>
   <Description>ModuleStreamNameAlias</Description>
   <Class>com.wowza.wms.plugin.streamnamealias.ModuleStreamNameAlias</Class>
</Module>
```

4. In the same Application.xml add the following property to the application level properties at the bottom of the file:

```
<Properties>

   <!-- ModuleStreamNameAlias Properties -->
   <Property>
      <Name>streamNameAliasFile</Name>
      <Value>${com.wowza.wms.context.VHostConfigHome}/conf/StreamNameAliasMap.txt</Value>
   </Property>
   <Property>
      <Name>streamNameAliasPathDelimiter</Name>
      <Value>/</Value>
   </Property>
   <Property>
      <Name>streamNameAliasNameDelimiter</Name>
      <Value>=</Value>
   </Property>
   <Property>
      <Name>streamNameAliasDebug</Name>
      <Value>true</Value>
      <Type>Boolean</Type>
   </Property>

   <!-- ModuleSecureURLParams Properties -->
   <Property>
      <Name>secureTokenSharedSecret</Name>
      <Value>#ed%h0#w@1</Value>
   </Property>

</Properties>
```

5. Edit "[install-dir]/conf/StreamNameAliasMap.txt" and remove or comment out the default rule (*=${Stream.Name}) and add an entry for each SHOUTcast URL you wish to allow to be re-streamed through this application. For example:

```
station1=http://192.168.1.7/station1
station2=http://192.168.2.63/station2
```

You will need to integrate the client side SecureToken response into your Flash player code.

RTSP/RTP Re-Streaming

This example illustrates how to protect live streaming when re-streaming an RTSP/RTP stream. Basic configuration:

```
Stream type: rtp-live
Add On Packages: SecureToken, StreamNameAlias
RTP/Authentication/Method: digest
rtp.password file: empty
StreamNameAliasMap.txt: entry for each RTSP/RTP URL
```

SecureToken and StreamNameAlias are needed to protect this form of re-streaming. You will need to download and install the StreamNameAliasPackage from here:

http://www.wowzamedia.com/forums/showthread.php?t=1505

The steps are to create an application named "rtsp" are:

1. Create the folder "[install-dir]/applications/rtsp"

2. Create the folder "[install-dir]/conf/rtsp" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file, change the Streams/StreamType to "rtp-live" , verify the RTP/Authentication/Method is set to "digest" and add the following <Module> definitions as the last entries in the <Modules> list (the order of these two modules is important):

```
<Module>
      <Name>ModuleSecureURLParams</Name>
      <Description>ModuleSecureURLParams</Description>
      <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
<Module>
   <Name>ModuleStreamNameAlias</Name>
   <Description>ModuleStreamNameAlias</Description>
   <Class>com.wowza.wms.plugin.streamnamealias.ModuleStreamNameAlias</Class>
</Module>
```

4. In the same Application.xml add the following property to the application level properties at the bottom of the file:

```
<Properties>

   <!-- ModuleStreamNameAlias Properties -->
   <Property>
      <Name>streamNameAliasFile</Name>
      <Value>${com.wowza.wms.context.VHostConfigHome}/conf/StreamNameAliasMap.txt</Value>
   </Property>
   <Property>
      <Name>streamNameAliasPathDelimiter</Name>
      <Value>/</Value>
   </Property>
   <Property>
      <Name>streamNameAliasNameDelimiter</Name>
      <Value>=</Value>
   </Property>
   <Property>
      <Name>streamNameAliasDebug</Name>
      <Value>true</Value>
      <Type>Boolean</Type>
   </Property>

   <!-- ModuleSecureURLParams Properties -->
   <Property>
      <Name>secureTokenSharedSecret</Name>
      <Value>#ed%h0#w@1</Value>
   </Property>

</Properties>
```

5. Edit "[install-dir]/conf/StreamNameAliasMap.txt" and remove or comment out the default rule (*=${Stream.Name}) and add an entry for each RTSP/RTP URL you wish to allow to be re-streamed through this application. For example:

```
stream1=rtsp://192.168.1.7/stream1
stream2=rtsp://192.168.2.63/stream2
```

You will need to integrate the client side SecureToken response into your Flash player code.

Example Players

The "Media Security" package includes several different examples of how to integrate SecureToken support into your Flash player ActionScript code. The "client" folder contains ActionScript 3.0 examples and the "clientAS2" folder contains ActionScript 2.0 examples. These examples use the application name "securestreaming". To setup an application to test these examples follow the steps:

1. Create the folder "[install-dir]/applications/securestreaming"

2. Create the folder "[install-dir]/conf/securestreaming" and copy "[install-dir]/conf/Application.xml" into this new folder.

3. Edit the newly copied "Application.xml" file and add the following <Module> definition as the last entry in the <Modules> list:

```
<Module>
     <Name>ModuleSecureURLParams</Name>
     <Description>ModuleSecureURLParams</Description>
     <Class>com.wowza.wms.plugin.security.ModuleSecureURLParams</Class>
</Module>
```

4. Add the following property to the application level properties at the bottom of the "Application.xml" file (be sure to add this property to the correct properties section at the bottom of the file):

```
<Properties>
  <Property>
     <Name>secureTokenSharedSecret</Name>
     <Value>#ed%h0#w@1</Value>
  </Property>
</Properties>
```